$\Rightarrow (A+B+(C\cdot\bar{C}))\cdot(\bar{A}+C+(B\cdot\bar{B}))$

$\Rightarrow (A+B)(\bar{A}+C)$ ✓

4. $A\bar{B}+BC+AC$

$\Rightarrow A\bar{B}(C+\bar{C})+BC(A+\bar{A})+AC(B+\bar{B})$

$\Rightarrow A\bar{B}C+A\bar{B}\bar{C}+ABC+\bar{A}BC$

$\Rightarrow A\bar{B}(C+\bar{C})+(A+\bar{A})BC$

$\Rightarrow A\bar{B}+BC\cdot$ ✓

continue...

0 01000 1011 0 (No need of addition)

00100 0101 1 (1) shift

10111

0 11011 0101 1 shr

01101 1010 1 (0)

∴ Hence proved.

Redundancy Theorem:

- It works only on 3 variables:
- Rules:
  - ⇒ Each variable should repeat twice in a exp.
  - ⇒ One variable should be complemented.
  - ⇒ Take the complemented variables.

Ex: 1. $AB + B\bar{C} + AC = Y$

Sol: Repeated twice ✓

Complemented $(C - \bar{C})$ ✓

$\therefore Y = B\bar{C} + AC$

$AB(C+\bar{C}) + B\bar{C}(A+\bar{A}) + AC(B+\bar{B})$

$\Rightarrow ABC + AB\bar{C} + A\bar{B}C + \bar{A}B\bar{C}$

$\Rightarrow B\bar{C}(A+\bar{A}) + AC(B+\bar{B})$

$\Rightarrow B\bar{C} + AC$ ✓

2. $\bar{A}\bar{B} + A\bar{C} + B\bar{C} = Y$

Sol: $\bar{A}\bar{B} + A\bar{C}$

3. $(A+B)\cdot(\bar{A}+C)\cdot(B+C) = Y$

Sol: $(A+B)(\bar{A}+C)$

4. $A\bar{B} + BC + AC$

Sol: $A\bar{B} + BC$

2. $\bar{A}\bar{B}(C+\bar{C}) + A\bar{C}(B+\bar{B}) + B\bar{C}(A+\bar{A})$

$\Rightarrow \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} + AB\bar{C} + A\bar{B}\bar{C}$

$\Rightarrow \bar{C}(\quad + \quad) + \bar{A}\bar{B}(C+\bar{C})$

$\Rightarrow \bar{A}C + \bar{A}\bar{B}$ ✓

3. $(A+B)\cdot(\bar{A}+C)\cdot(B+C)$

$\Rightarrow (A+B+C\cdot\bar{C})\cdot$

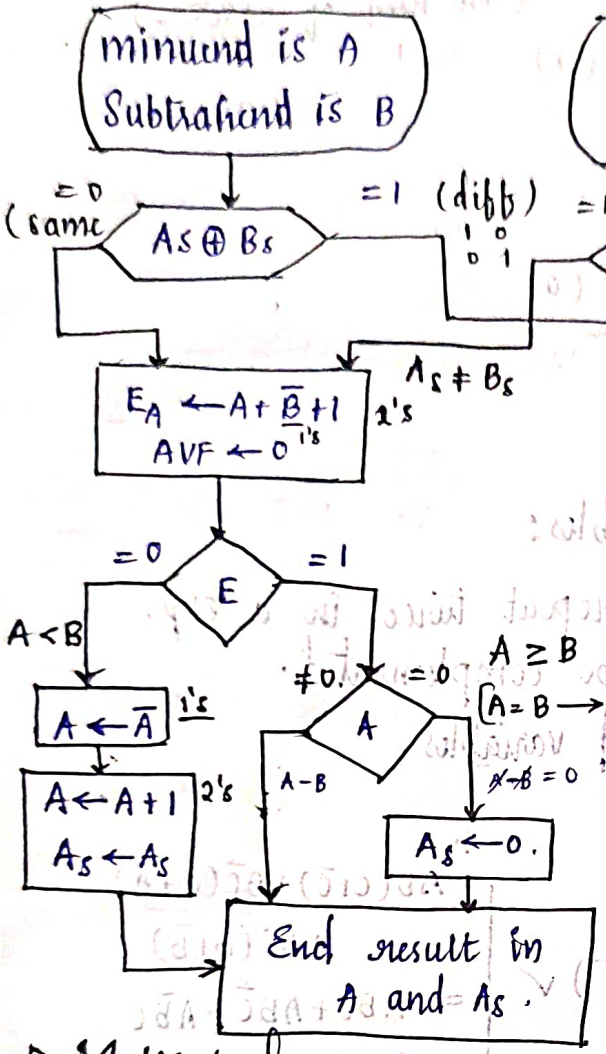
$(\bar{A}+C+B\cdot\bar{B})\cdot$

$(B+C+A\cdot\bar{A})$

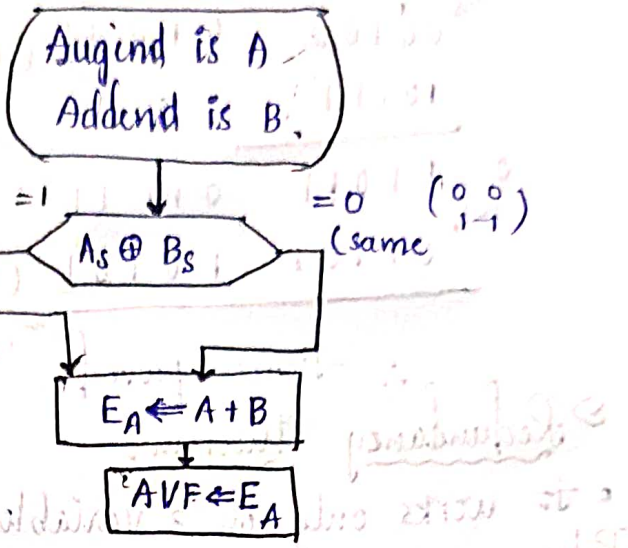
$\Rightarrow (A+B+C)\cdot(A+B+\bar{C})\cdot$

$(\bar{A}+B+C)\cdot(\bar{A}+\bar{B}+C)$

* Subtract



Add



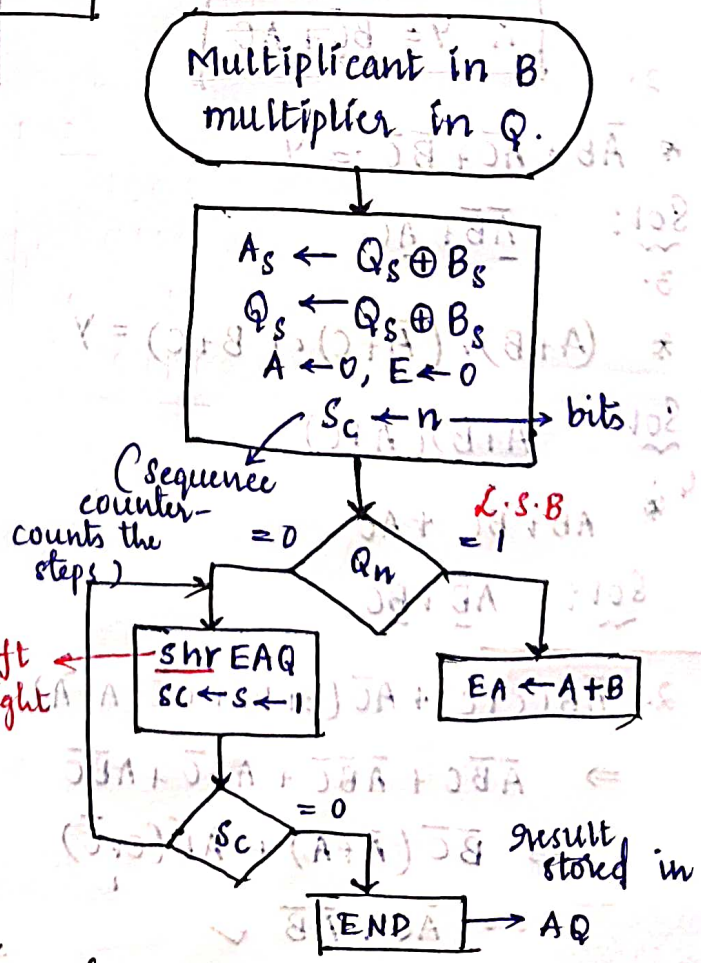
* Multiplication:

i) 10111 B 23
x 10011 Q x 19

10111
10111+
00000+
00000+
10111+
110101011 = 437 ✓

Calculation:

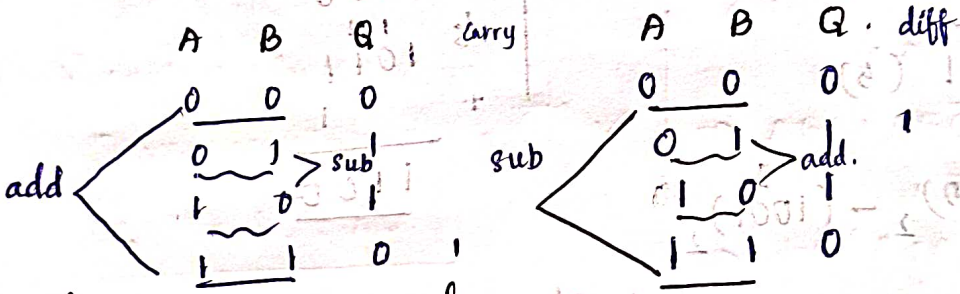
E	A	Q	SC.	
0	00000	10011	5	
+	10111			L·S·B
0	10111	10011		→ Lost (5)
	01011	11001		shr
+	10111			
1	00010	11001		→ Lost (4)
	10001	01100		shr
	01000	10110		L·S·B → 0 (no need to perform addition) → Lost (3)



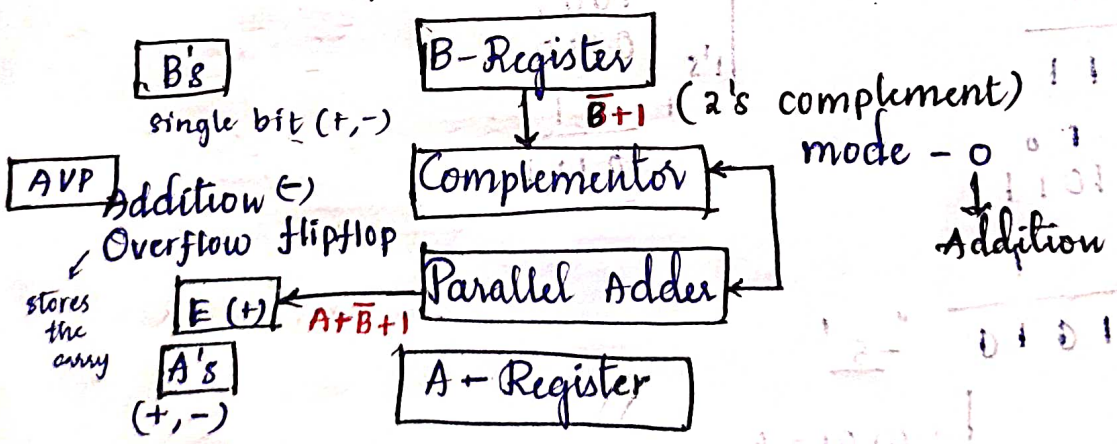
⊙ Addition & Subtraction Algorithm:

Operation	Add	A > B	A < B	A = B
(+A) + (+B)	A + B			
(+A) + (−B)		+(A−B)	−(B−A)	+(A−B)
(−A) + (+B)		−(A−B)	+(B−A)	+(A−B)
(−A) + (−B)	−(A+B)			
(+A) − (+B)		+(A−B)	−(B−A)	+(A−B)
(+A) − (−B)	A+B			
(−A) − (+B)	−(A+B)			
(−A) − (−B)		−(A−B)	+(B−A)	+(A−B)

* XOR : Adder Substractor



⊙ Hardware Implementation :



2) $(0011)_2 - (0101)_2$

Sol:

0011

2's : 1011

carry 0 | 1110

1110

− 1

1101 "−2"

−ve

2's

0101

1's 1010

+ 1

2's 1011

⌈ 0010

Ex: i) $(1001)_2 - (0100)_2$

1001

2's C : 1100

carry 1 | 0101 (5)

2's

D100

1's 1011

+ 1

1100

ii) $(0100)_2 - (1001)_2$

0100

2's C : 0111

carry 0 | 1011

1011

− 1

1010 "−5"

−ve

2's

1001

1's 0110

+ 1

0111

⌈ 0101 = 5 ⌋

* 3. COMPUTER ARITHMETIC *

Substraction by using complements :

- using 1's Complement: $A - B = A + (-B)$

1. Find the 1's complement of the no. to be substracted (B). — subtrahend.
2. Perform addition. $(A + \overline{B})$
3. If the final carry is '1' add that 1 to the result.
4. If the final carry is '0' the result is in the -ve form.

Ex: 1) $(0101)_2 - (0011)_2$ (5, 3)

Sol: 0101

1's complement: 1100

\+ 1 | 0001

carry: 1

\+ 0010 (2).

2) $(0011)_2 - (0101)_2$ (3, 5)

0011

1's complement: 1010

\+ 0 | 1101

carry ← 0

-ve form

0010 (2)

↳ "-2"

- using 2's complement:

1. Find the 2's complement of the no. to be substracted.
2. Perform Addition.
3. If the final carry is '1', the result is +ve.
4. If the final carry is '0', the result is -ve and substract 1 from the result.

Ex: 1) $(0101)_2 - (0011)_2$ (5, 3)

0101

2's C: 1101

carry 1 | 0010 (2)

2's complement

0011

1's C- 1100

\+ 1

2's C 1101

* Asynchronous

- In this, the output of 1st flipflop drives the clock to the next flipflop.
- Clk is not applied simultaneously.
- Circuit is simple for more no. of states.
- Speed is slow as clk is propagated through no. of states.

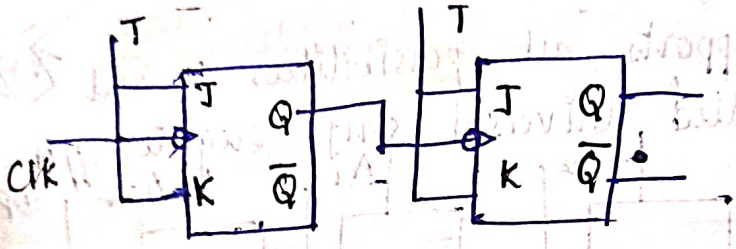
Synchronous

- There is no connection b/w output of 1st flipflop and clk of the next.
- Clk is connected simultaneously.
- Circuit becomes complicated as the no. of states ↑.
- Speed is high as clk is given at the same time.

Counters:

- Counters are used to count the no. of clock pulses in a sequential circuit.
- There are two types of Counters:
  1. Asynchronous / Ripple
  2. Synchronous.

* Asynchronous / Ripple Counter:



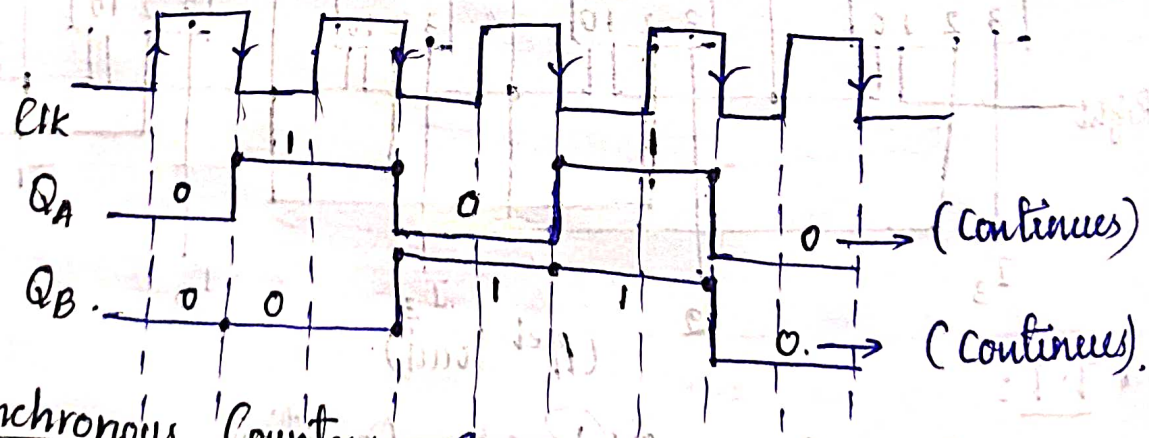
(Previous output will be the clk for the second)

- For n-bit counter, 2^n clock pulses will be generated.

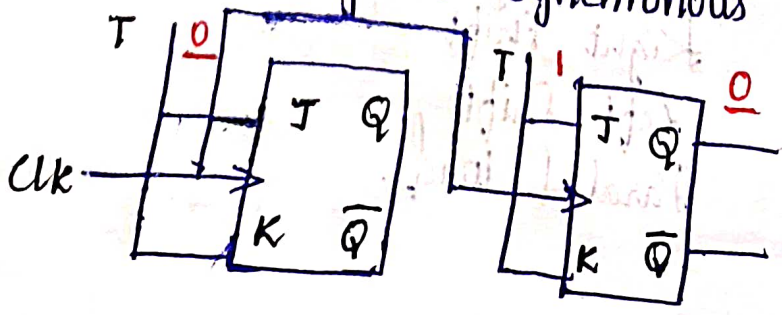
- T-T:

2^2

Clk	Q_B	Q_A
0	0	0
1	0	1
2	1	0
3	1	1

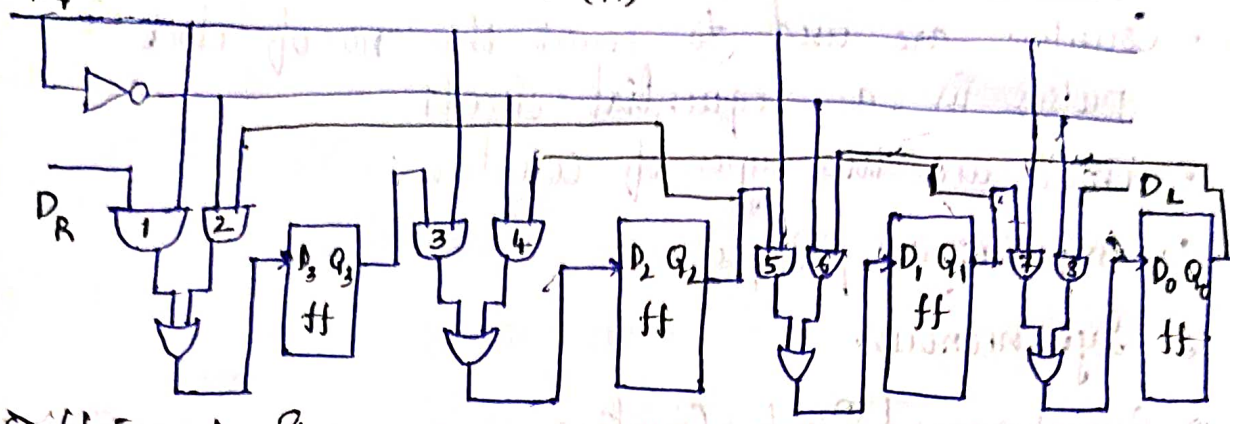


* Synchronous Counters: Same clk is applied simultaneously in synchronous counters.



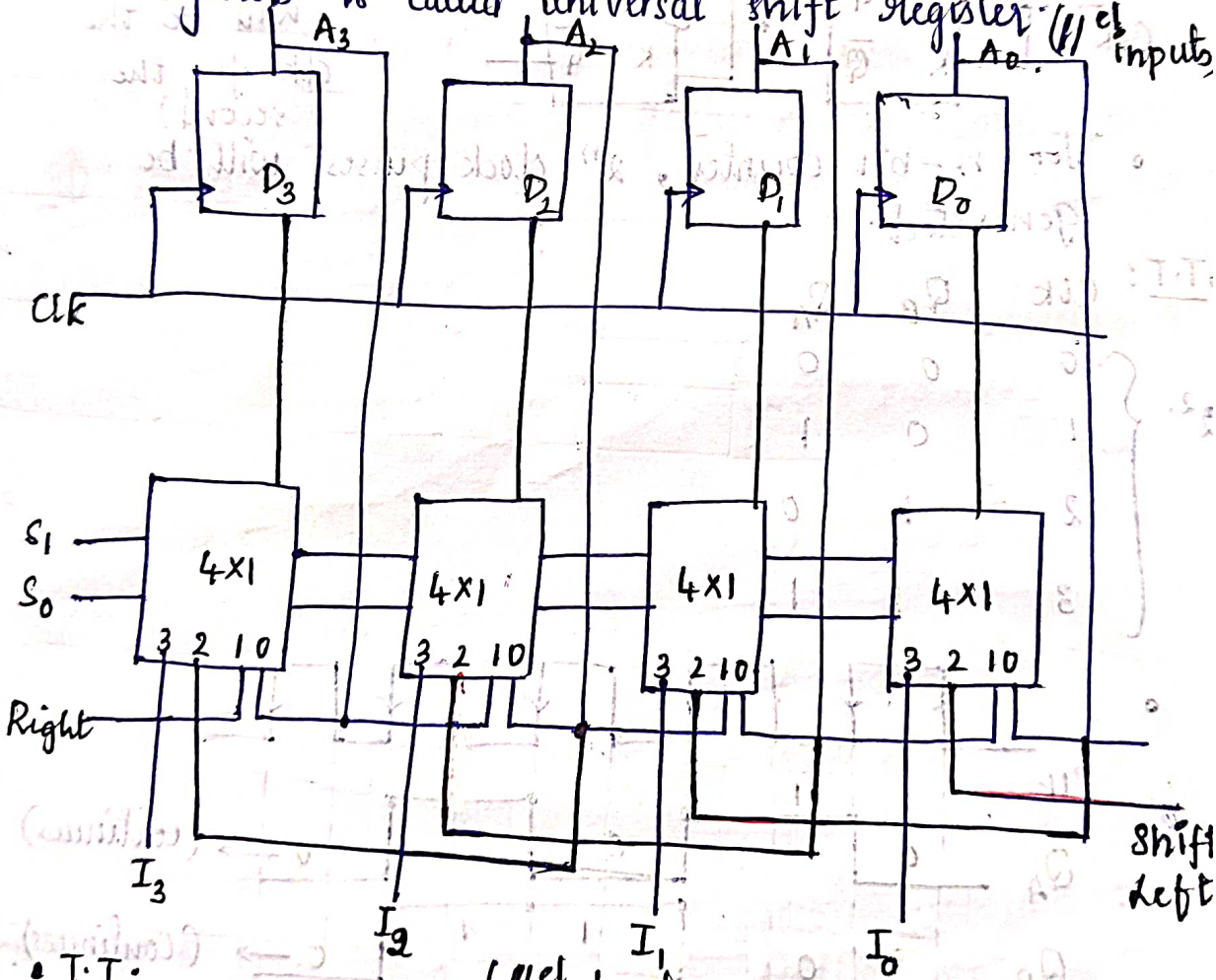
Clk	Q_B	Q_A
0	0	0
1	0	1
2	1	0

* Mode = 0 (L)
1 (R).



Universal Shift Register:

- A register that supports all possibilities of all the registers is called universal shift register. (11 el inputs)

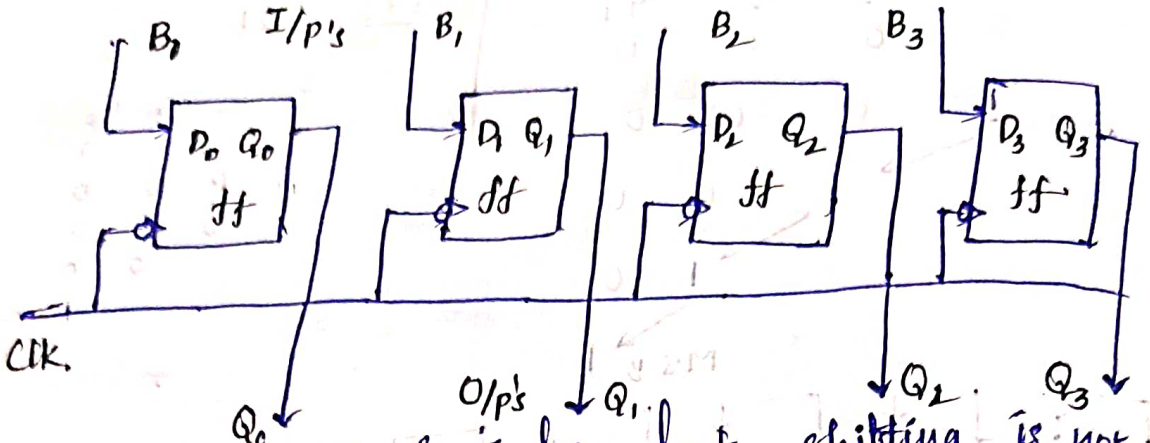


- T.T:

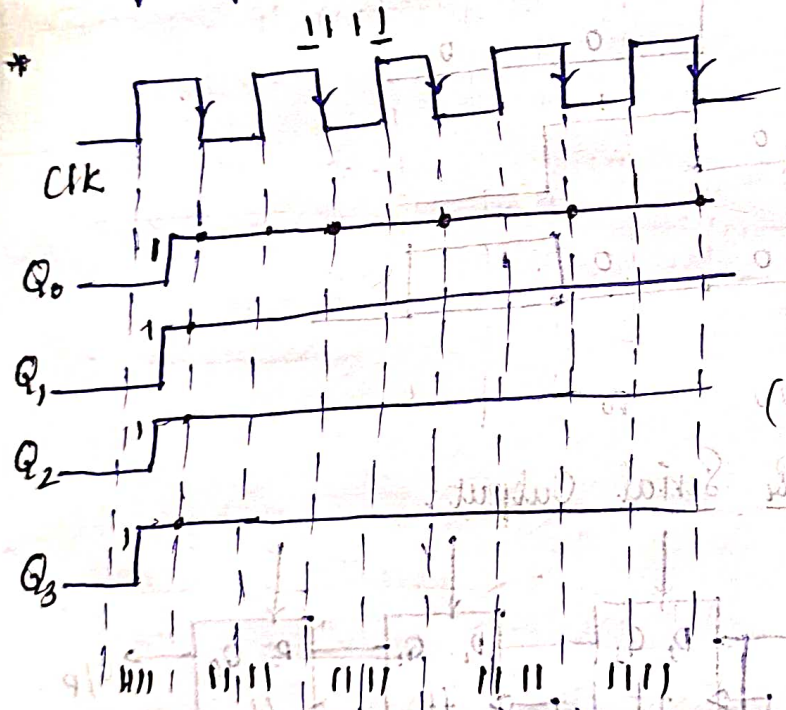
(11 el load)

	S_1	S_0	Register Operation
I_0	0	0	No change
I_1	0	1	Right Shift
I_2	1	0	Left Shift
I_3	1	1	Parallel load.

(v) Ilel Input & Output:

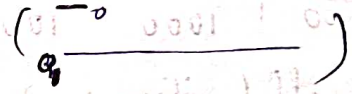


- As each output is independent, shifting is not required.
- It is a storage Register; No need to perform shift operation.



Q_n (previous output).

Ex:



⇒ Bi-directional Register:

- To perform left shift operation, do multiplication.

- To perform right shift operation, do division

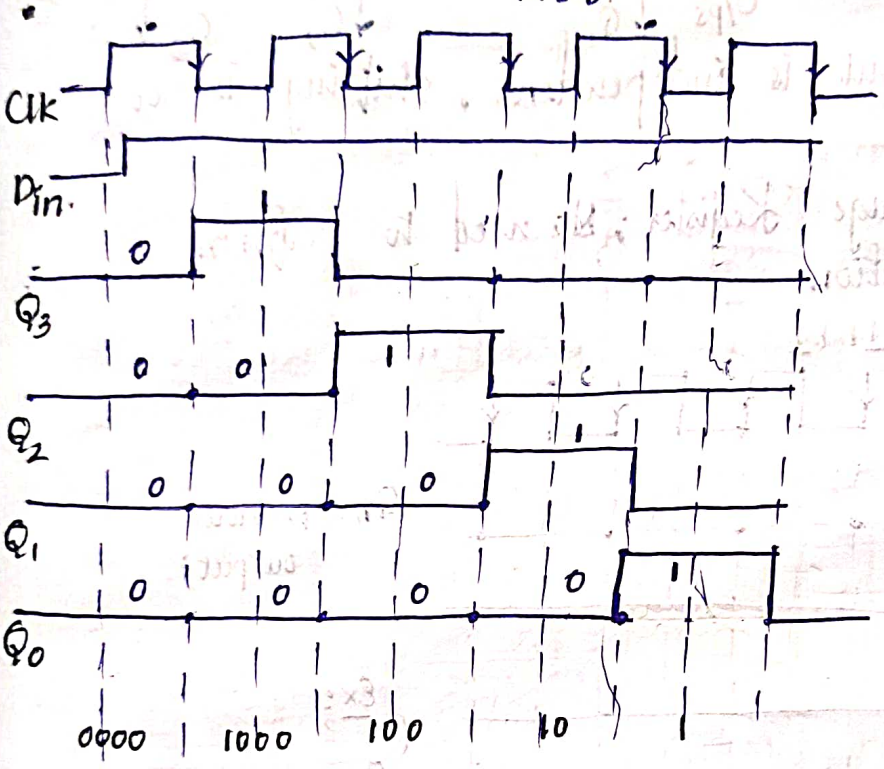
• T·T:

Clk	Q_3	Q_2	Q_2	Q_1
	0	0	0	0
	1	0	0	0
		1	0	0
			1	0
				1

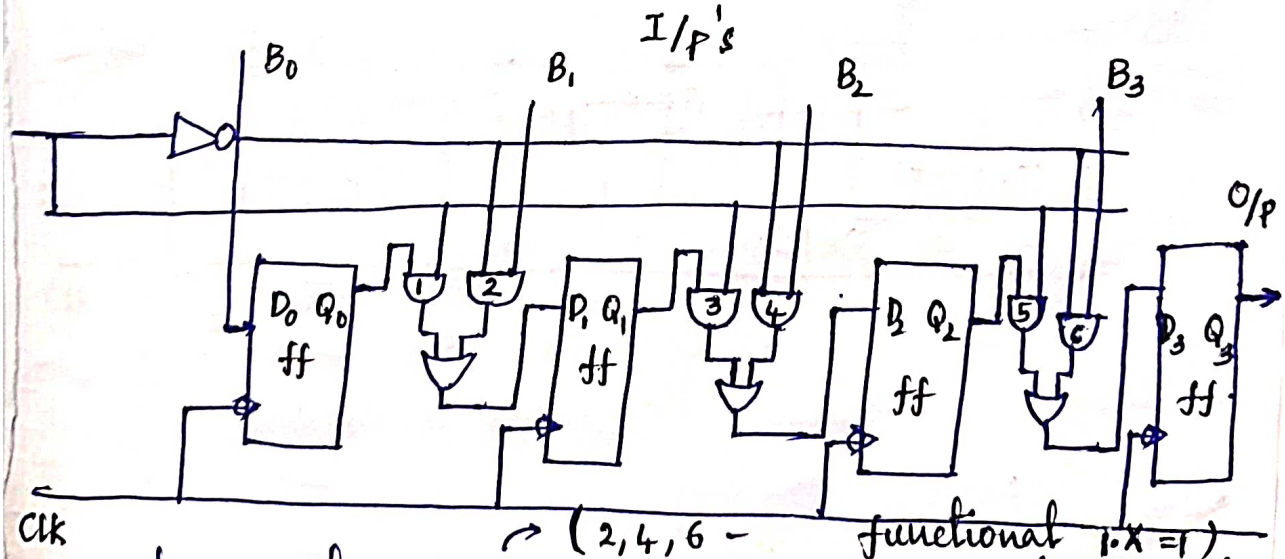
L·S·B → M·S·B

Ex: 1011

Clk	Q_3	Q_2	Q_1	Q_0
	0	0	0	0
	1	0	0	0
		1	0	0
			0	0
			0	1



⇒ Parallel Input & Serial Output:



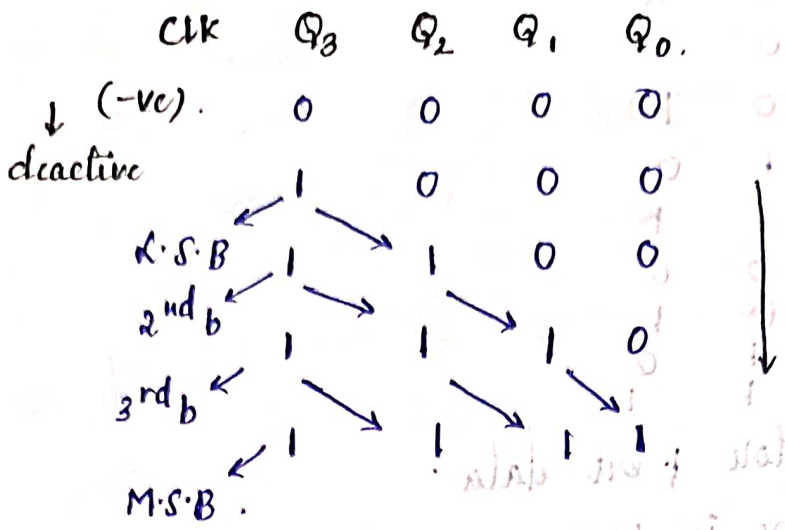
(2,4,6 - functional 1·X = 1).

• To load the data into the circuit mark shift = 0. (1,3,5 N-F 0·X = 0

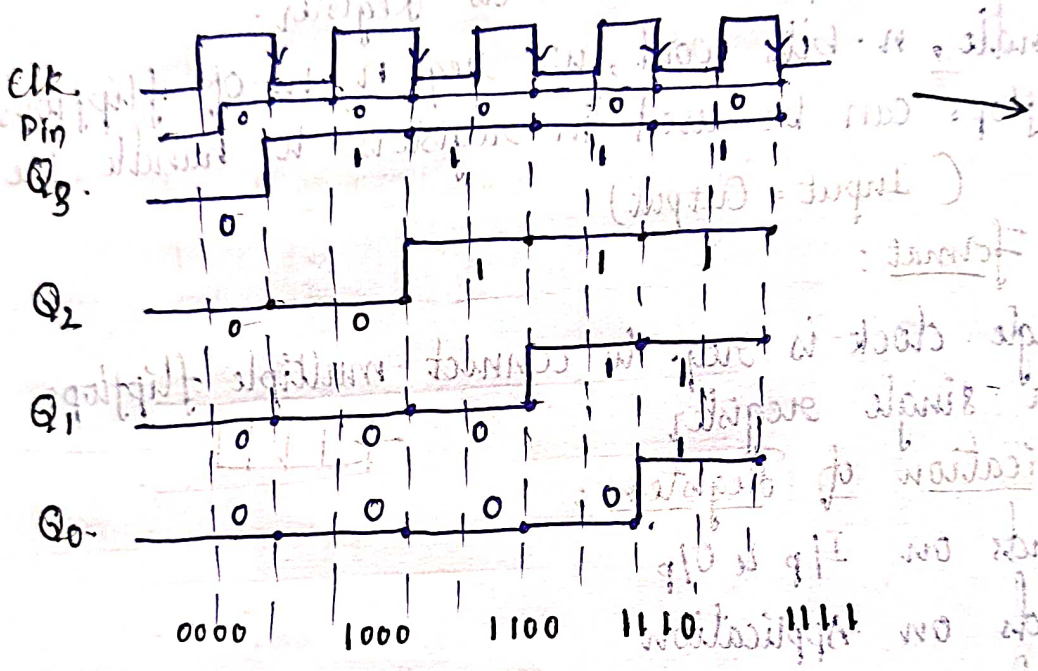
• To shift the data from one flip-flop to another mark shift = 1 (1,3,5 Functional) 1·X = 1

2,4,6 → Non functional 0·X = 0

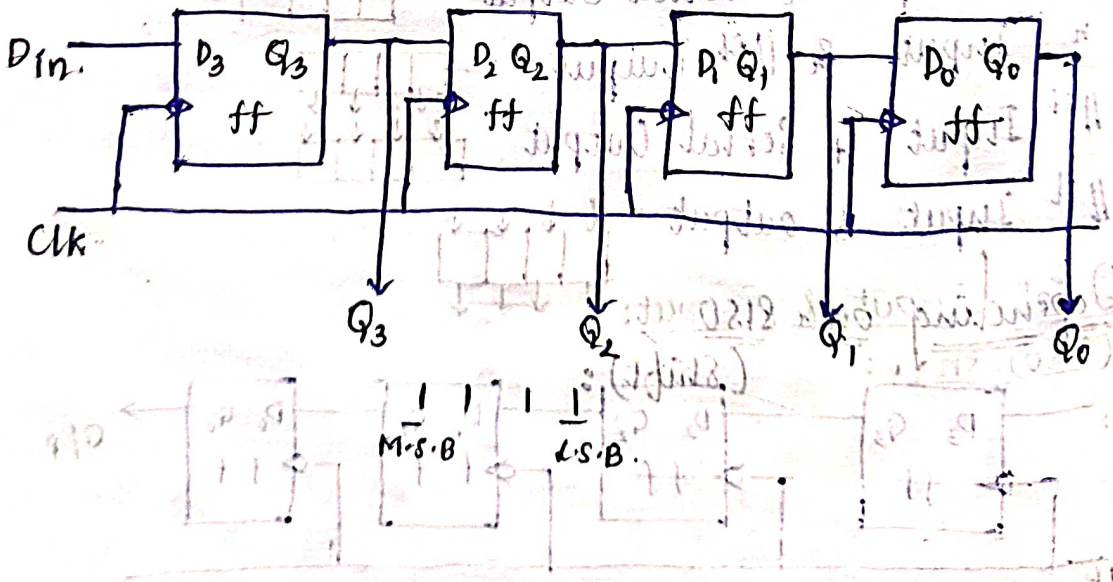
- T-T :



⇒



⇒ Serial Input & Ilel Output :



* Excitation Table:

Q_n	Q_{n+1}	J	K
0	0	0	0
0	0	0	1
0	1	1	0
0	1	1	1
1	1	0	0
1	0	0	1
1	1	1	0
1	0	1	1

⇒ Registers:

- A flipflop can store 1-bit data.
- Group of flipflops is known as Register.
- To handle, n-bits word, we req. n no-of flipflops.
- D-flipflops can be used in registers to handle the data. (Input = Output)
- Data Format:

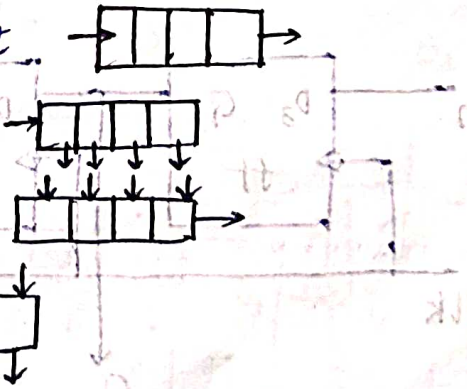
A single clock is req. to connect multiple flipflops in a single register

⇒ Classification of Registers:

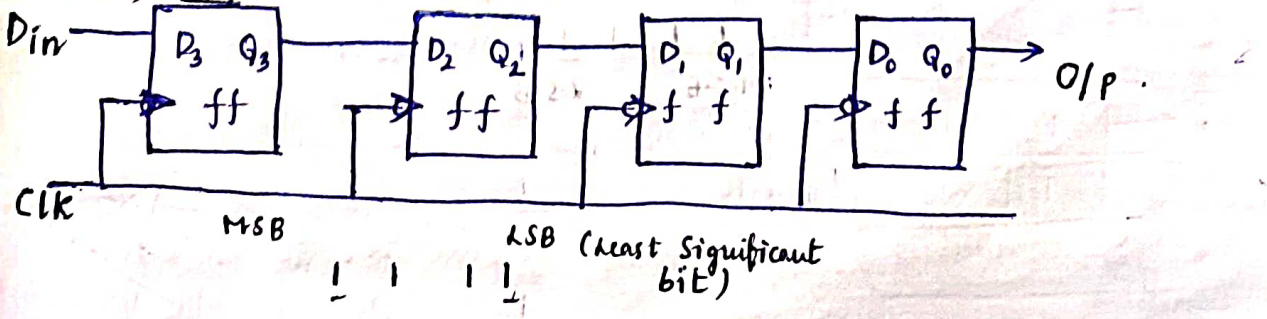
- Depends on I/p & O/p
- Depends on Application

⇒ Depending on I/p & O/p:

1. Serial Input & Serial Output.
2. n - Input & ||el Output.
3. ||el Input & Serial Output
4. ||el Input & output.



⇒ Serial Input & Output: (SISO) shift:



- **Excitation Table:**

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

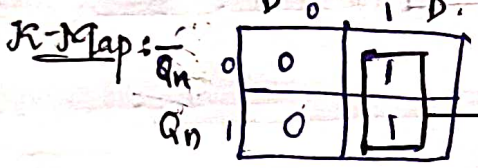
* **D-Flip-flop Characteristic table:**

*

Q_n	D	Q_{n+1}	
0	0	0	-0
0	1	1	-1
1	0	0	-2
1	1	1	-3

[from D-flip flop T.T].

K-Map:



$\therefore Q_{n+1} = D$

* **Excitation Table:**

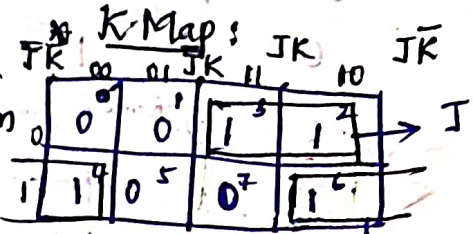
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

* **J-K Flipflop Characteristic Table:**

Q_n	J	K	Q_{n+1}	
0	0	0	0	
0	0	1	0	
0	1	0	1	
0	1	1	1	$(\bar{Q}_n)$
1	0	0	1	
1	0	1	0	
1	1	0	1	
1	1	1	0	$(\bar{Q}_n)$

[from J-K flipflop T.T]

* K-Map:



$\therefore Q_{n+1} = J + Q_n\bar{K}$

• T-T:

Clk	T	Q_{n+1}	$\bar{Q}_{n+1}$
0	X	Q_n (Memory)	
1	0	Q_n	
1	1 ($\frac{J=1}{K=1}$)		$\bar{Q}_n$

($T=1, J \& K = 1$)

(From J-K Flipflop T-T).

Characteristic Table:

Q_n	T	Q_{n+1}	
0	0	0	
0	1	1	} Active
1	0	1	} Active
1	1	0	

$\therefore Q_{n+1} = \bar{Q}_n T + Q_n \bar{T} = Q_n \oplus T$ (X-OR)

$$\boxed{\therefore Q_{n+1} = Q_n \oplus T}$$

Excitation Table:

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

(From Characteristic Table).

* These tables are used in counters and conversion of flipflops.

S-R Flip flop Characteristic Table:

•T-T:

Q_n	S	R	Q_{n+1}	
0	0	0	0	- 0
0	0	1	0	- 1
0	1	0	1	- 2
0	1	1	X	- 3
1	0	0	1	- 4
1	0	1	0	- 5
1	1	0	1	- 6
1	1	1	X	- 7

(From S-R Flipflop table).

(0,0) Excitation Table: unchanged 0, unchanged 0, change (X) 0

1,1: change (X) 0, unchanged 0

• K-Map:

Q_n	$\bar{S}\bar{R}$ (00)	$\bar{S}R$ (01)	SR (11)	$S\bar{R}$ (10)
Q_n 0	0 (0)	0 (1)	X (3)	1 (2)
Q_n 1	1 (4)	0 (5)	X (7)	1 (6)

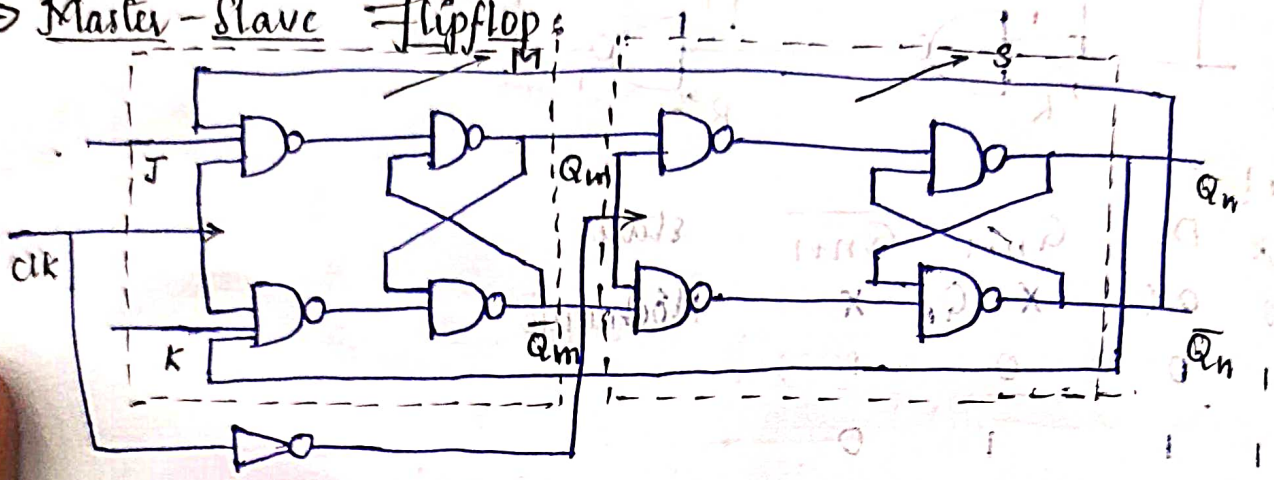
S

$Q_n\bar{R}$

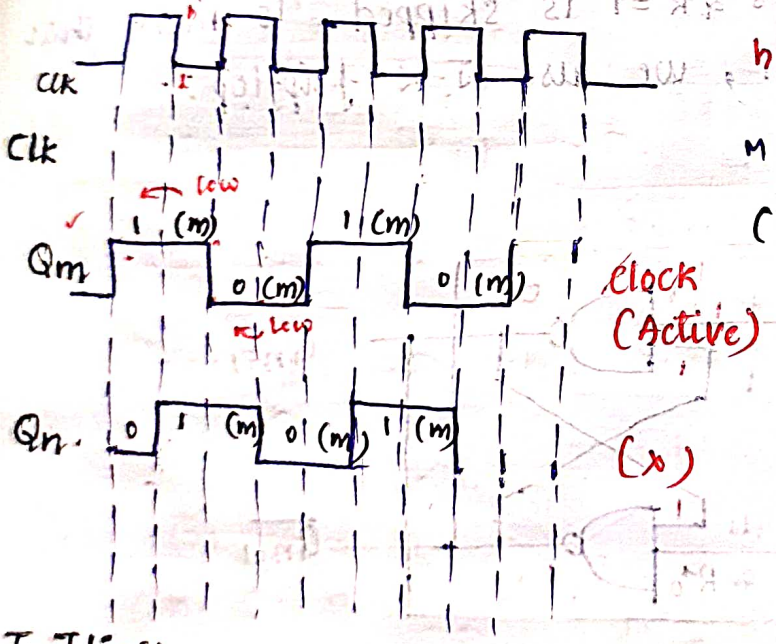
$$\Rightarrow \boxed{Q_{n+1} = S + Q_n\bar{R}}$$

- Its a race around condition
- Toggle state can be controlled & Race around condition can't be controlled
- The solution for this problem is Master-Slave Flipflop

⇒ Master-Slave Flipflop:

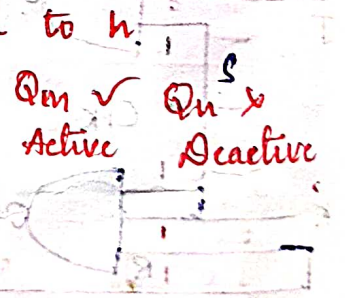


*

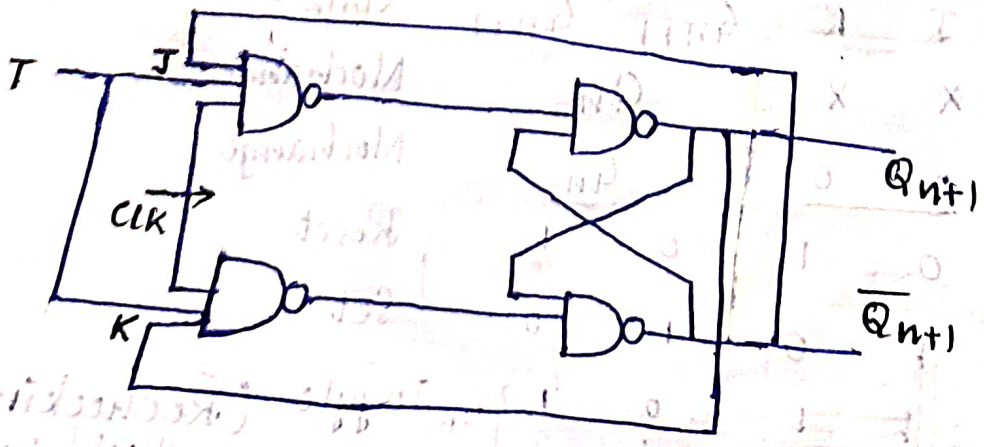


h to l
clk → 0.
M Q_m ✗ Q_n ✓
Deactive Active
(m) → memory state

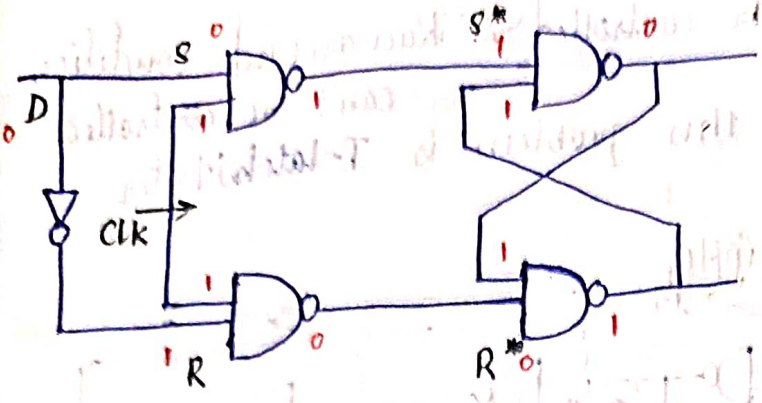
l to h
M Q_m ✓ Q_n ✗
Active Deactive



⇒ T-FlipFlop: (Toggle)



• If S & R = 1, the output is invalid. So we use D flipflops.

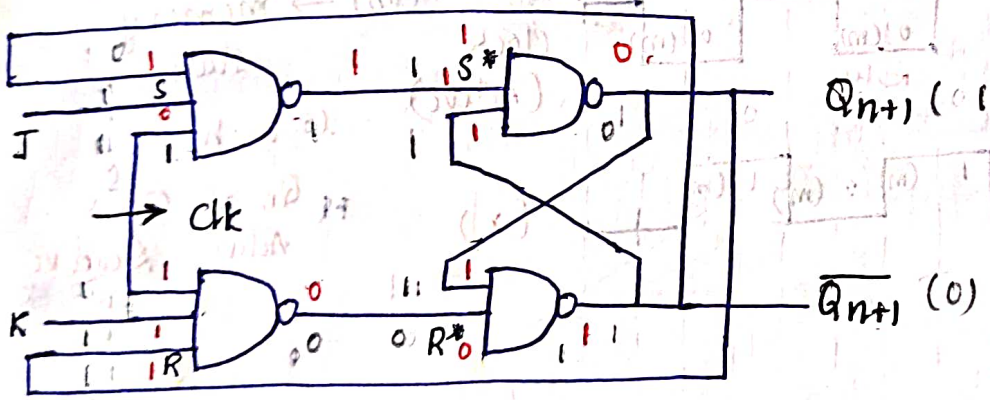


• T.T:

Clk	D	Q_{n+1}	$\overline{Q_{n+1}}$
0	0X	Q_n	
1	0	0	1 ✓
1	1	1	0

* In de-flipflop, D·S & R = 1 is skipped, to make that use of S = 1 & R = 1, we use J-K flipflop.

2 J-K flipflop:

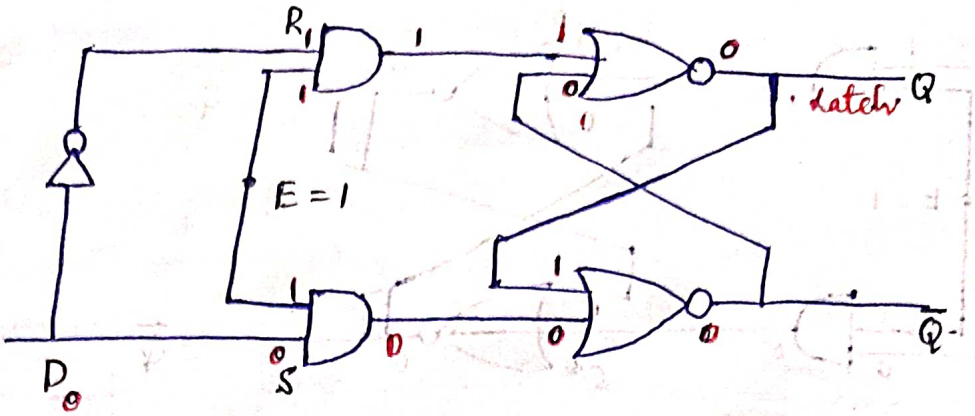


• T.T:

Clk	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	State
0	X	X	Q_n		No change
1	0	0	Q_n		No change
1	0	1	0	1	Reset
1	1	0	1	0	Set
→ 1	1	1	0	1	$\overline{Q_n}$ Toggle (Rechecking should be done)
			$\overline{Q_n}$	Q_n	

• When J & K = 1, the outputs of Q_{n+1} are 0 1 0 1 0 1
& $\overline{Q_{n+1}}$ are 1 0 1 0 1 0

* D-Latch [using NOR]:



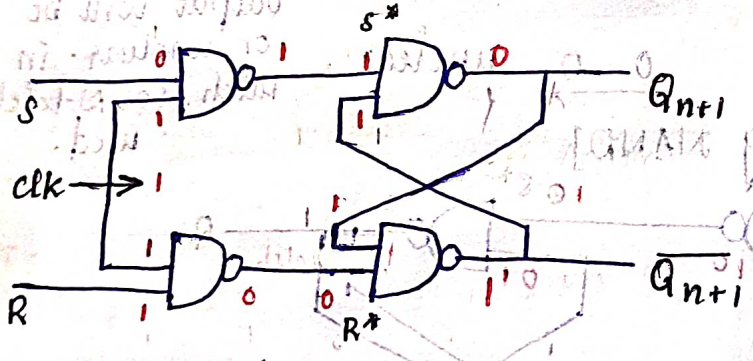
• T.T:

E	D	Q	$\overline{Q}$
0	X	Latch	
1	0	0	1
1	1	1	0

⊙ Flip flops:

* It is a storage device that stores 1 bit data.
* It contains clock signal.
* S-R (+ve clock) Flipflop:

[→↑ +ve clock signal
→↓ -ve clock signal]



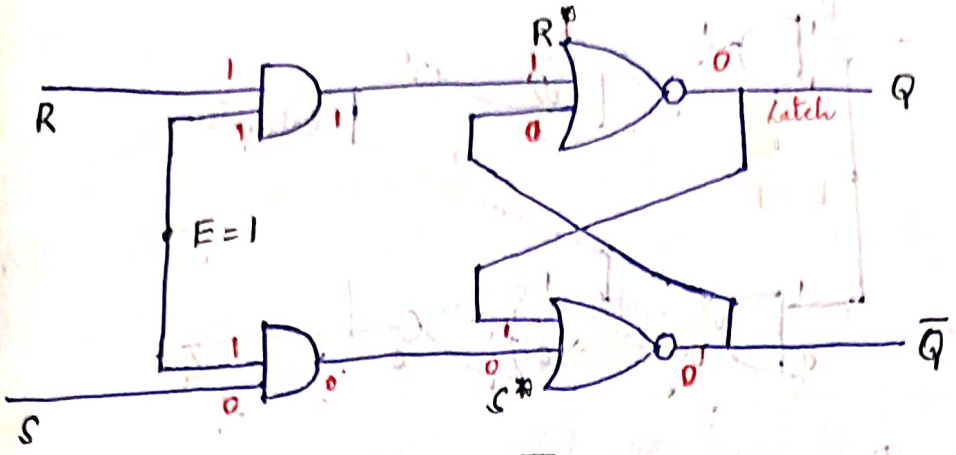
• T.T:

clk	S	R	Q_{n+1}	$\overline{Q}_{n+1}$	State
0	X	X	Q_n		No change
1	0	0	Q_n		No change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	X	X	Invalid

(or) there is another way to get flipflop T by S-R latch Nand-gate)

$$\begin{cases} S^* = \overline{S \cdot clk} = \overline{S} + \overline{clk} \\ R^* = \overline{R \cdot clk} = \overline{R} + \overline{clk} \end{cases}$$

* Gated SR Latch [By two and (AND) gates] :

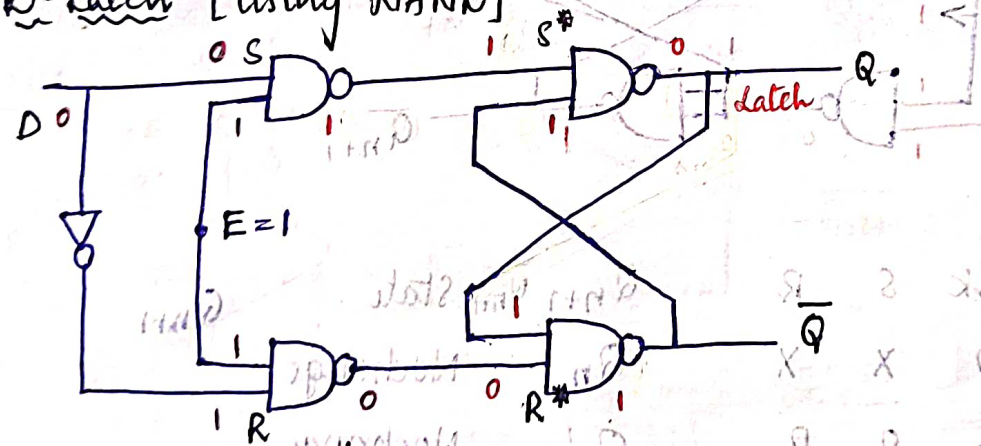


T.T:

E	S	R	Q	$\bar{Q}$
0	0	0	Latch	
0	0	1	Latch	
0	1	0	Latch	
0	1	1	Latch	
1	0	0	Latch	
1	0	1	0	1 ✓
1	1	0	1	0
1	1	1	0	0 unclear

* Note: If S & R = 1 then output will be invalid or unclear. in S-R latch, so D-latch is used.

* D-Latch [using NAND]



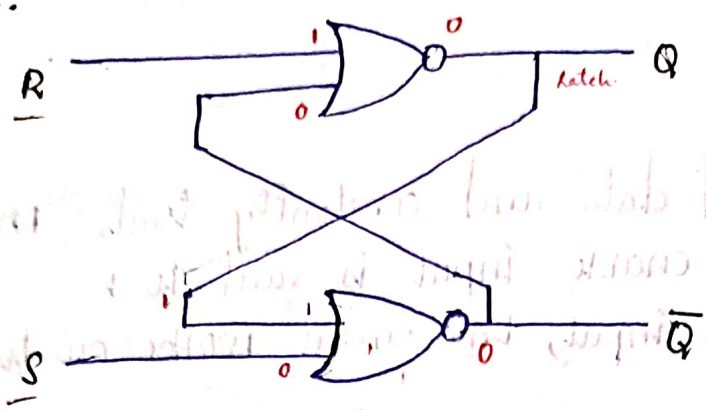
* T.T: (NOR always for R)

E	D	Q	$\bar{Q}$
0	X (don't care)	Latch	
1	0	0	1
1	1	1	0

[* E = 0 → Enable ↑ So, always latch]

* In D-Latch, output is same as input.

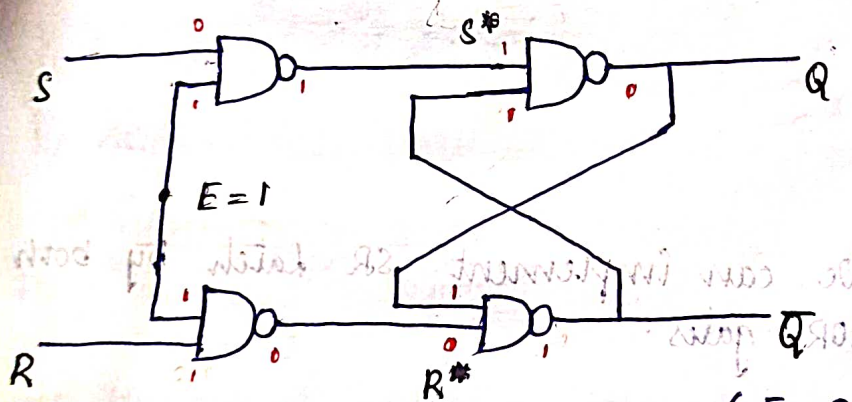
• NOR:



T.T:

S	R	Q	$\bar{Q}$
0	0	Latch	
0	1	0	1
1	0	1	0
1	1	0	0 → unclear.

* Gated S-R Latch by using NAND:



• T.T:

E	S	R	Q	$\bar{Q}$	
Latch	0	0	0	Latch	
	0	0	1	Latch	
	0	1	0	Latch	
	0	1	1	Latch	
	1	0	0	Latch	
	1	0	1	0	1
	1	1	0	1	0
	1	1	1	0	0 → unclear

(E=0. Circuit doesn't function, so it stores previous output).

- It has 2 states :
  1) Low (off)
  2) High (On)
- It changes the stored data and constantly trails the inputs when the enable input is said to 1.
- Based on the enable input, the circuit works on two states :

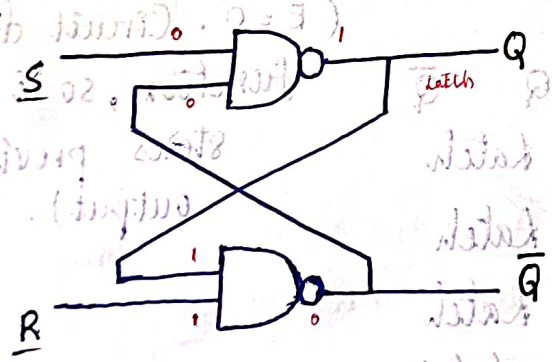
⇒ When it is high, then the both inputs are low.

⇒ " low, " high.

⇒ Types of Latches :
1) S-R Latch (set-Reset)
2) Gated S-R Latch
3) D - Latch
4) Gated D-Latch
5) J-K Latch
6) T Latch

⇒ S-R Latch : We can implement SR Latch by both NAND and NOR gates.

- NAND :



T.T :

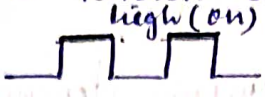
S	R	Q	$\overline{Q}$
0	0	Latch (stores 1-bit)	
0	1	1	0
1	0	0	1
1	1	0	0 unclear.

⇒ In S-R Latch, the outputs of using NAND & NOR-gates are complement to each other.

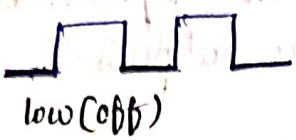
⇒ If S=1 & R=1 then output is invalid / unclear.

⇒ Level triggering:

- A circuit is activated when the logic level is high, it is known as +ve level triggering.

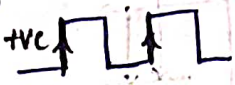


- A circuit is activated when the logic level is low, it is known as -ve level triggering.

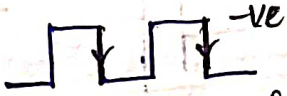


⇒ Edge Triggering:

- The transition from low to high is +ve edge triggering.



- The transition from high to low is -ve edge triggering.



⇒ Types of Sequential Circuits:

i) Synchronous S.C's.

ii) Asynchronous S.C's.

* Synchronous S.C's	* Asynchronous S.C's.
• They are easy to design.	• They are difficult to design.
• In this, a clocked flip-flop acts as a memory element.	• In this, a latch acts a memory element.
• In this, it takes too long to get the result.	• In this, we get faster result as there is no clock available
• In this, the state of the memory element is affected only at active edge of the clock if input is changed. (acts on edge triggering)	• In this, state of memory element is changed as soon as the input is changed. (acts on level triggering)

⇒ Latch: It is a special type of logical circuit, which is used to store the previous output.

- It stores 1 bit until the device sets to 1.

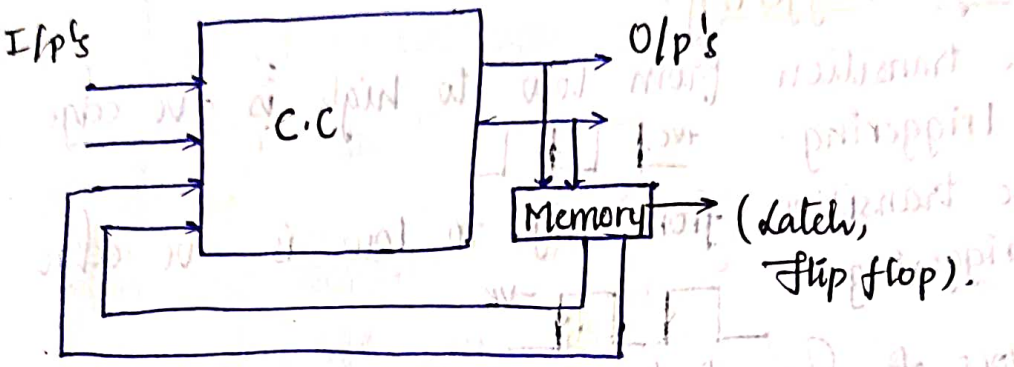
Sequential Circuits:

- Combinational Circuits:

⇒ In combinational circuits, present outputs depends upon present inputs.

⇒ In sequential circuits, present outputs depends both on present inputs as well as previous outputs.

B.D:



Eg: Counters.

- In sequential circuits, memory can be a latch or flip flops.

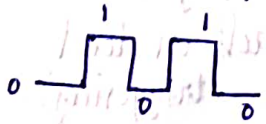
Diff b/w Combinational & Sequential circuits:

* C.C (Def)	* S.C. (Def).
1) Memory is not required.	1) Memory is req to store the previous outputs.
2) Feedback path is not req.	2) Feedback path is req.
3) There is no clock signal	3) There is clock signal requirement in S.C.

- Clock Signal (Clk):

⇒ It decides the time period of output.

⇒ It can be represented using square wave forms.



⇒ There are two types:

1. Level Triggering
2. Edge Triggering.

- T.T:

D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

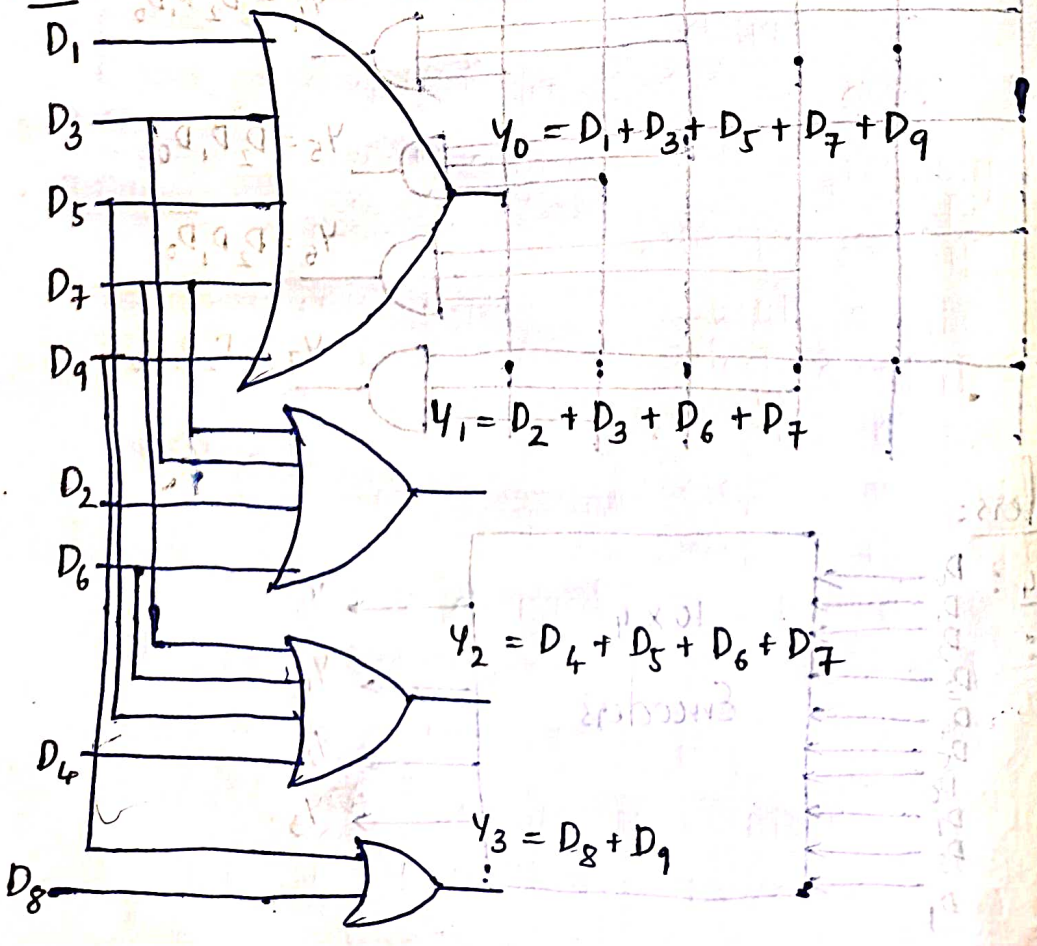
Y_0 : $D_1 + D_3 + D_5 + D_7 + D_9$.

Y_1 : $D_2 + D_3 + D_6 + D_7$.

Y_2 : $D_4 + D_5 + D_6 + D_7$.

Y_3 : $D_8 + D_9$.

- C.D:



D_2	D_1	D_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$Y_0 : \overline{D_2}\,\overline{D_1}\,\overline{D_0}$

$Y_1 : \overline{D_2}\,\overline{D_1}\,D_0$

$Y_2 : \overline{D_2}\,D_1\,\overline{D_0}$

$Y_3 : \overline{D_2}\,D_1\,D_0$

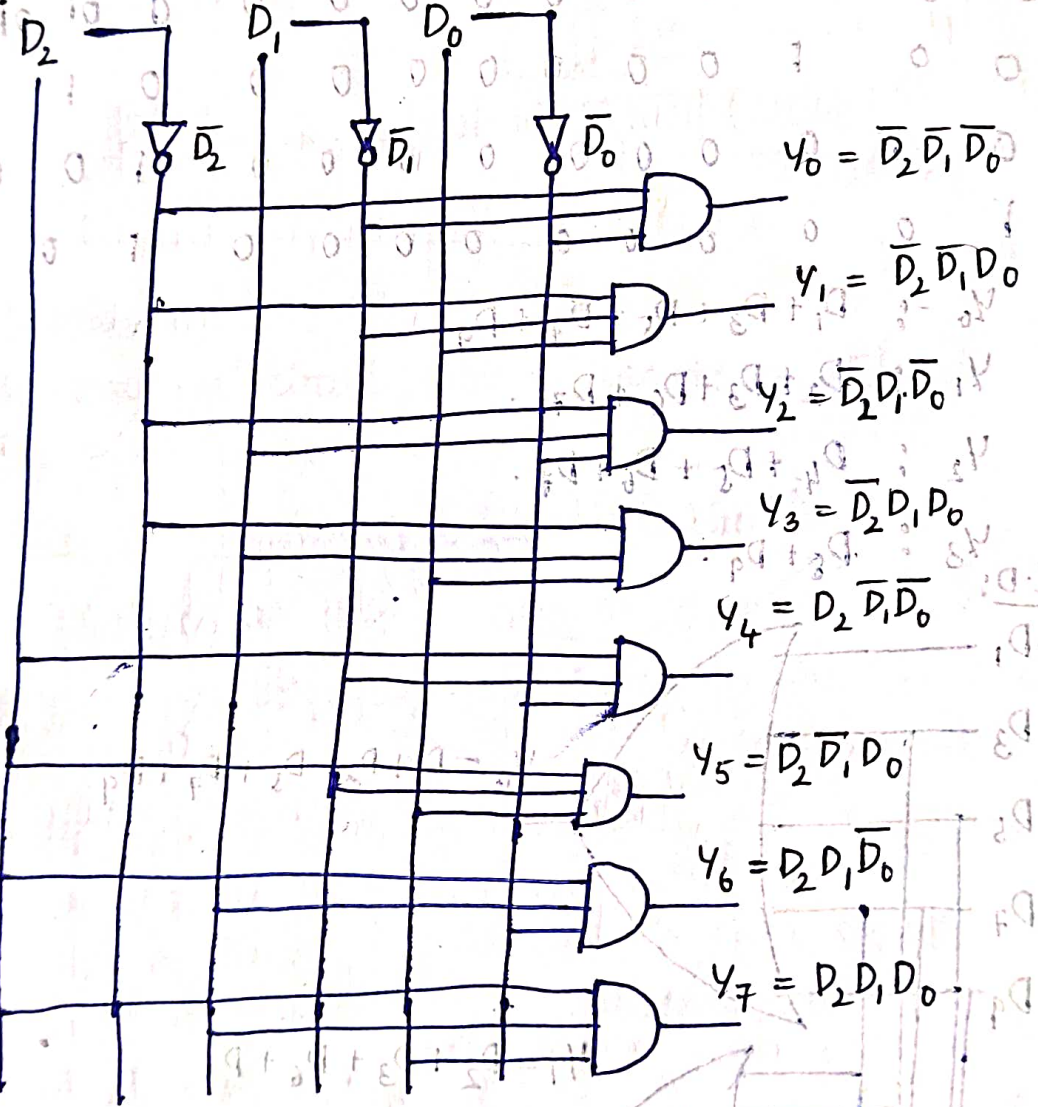
$Y_4 : D_2\,\overline{D_1}\,\overline{D_0}$

$Y_5 : D_2\,\overline{D_1}\,D_0$

$Y_6 : D_2\,D_1\,\overline{D_0}$

$Y_7 : D_2\,D_1\,D_0$

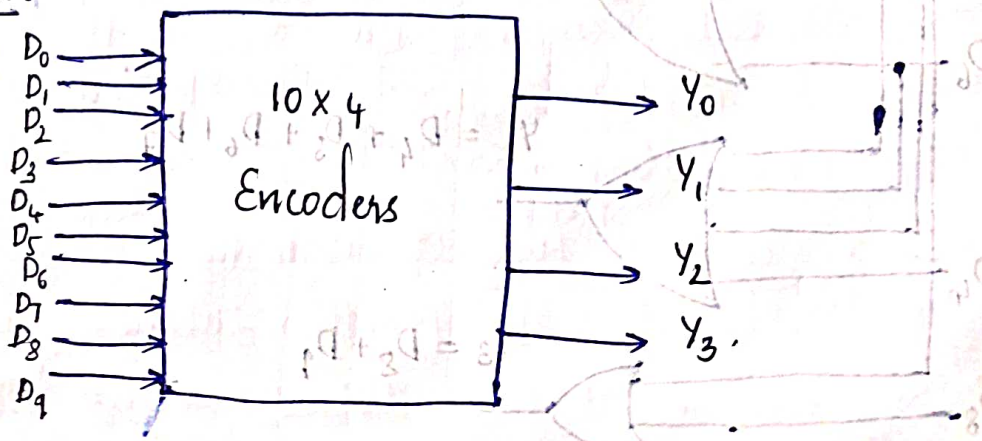
• C.D:



⇒ Encoders:

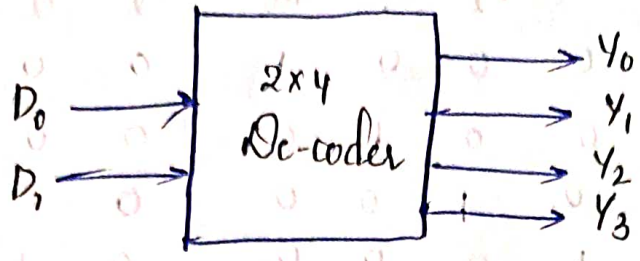
* 10 x 4 :

• B.D:



* 2x4 :

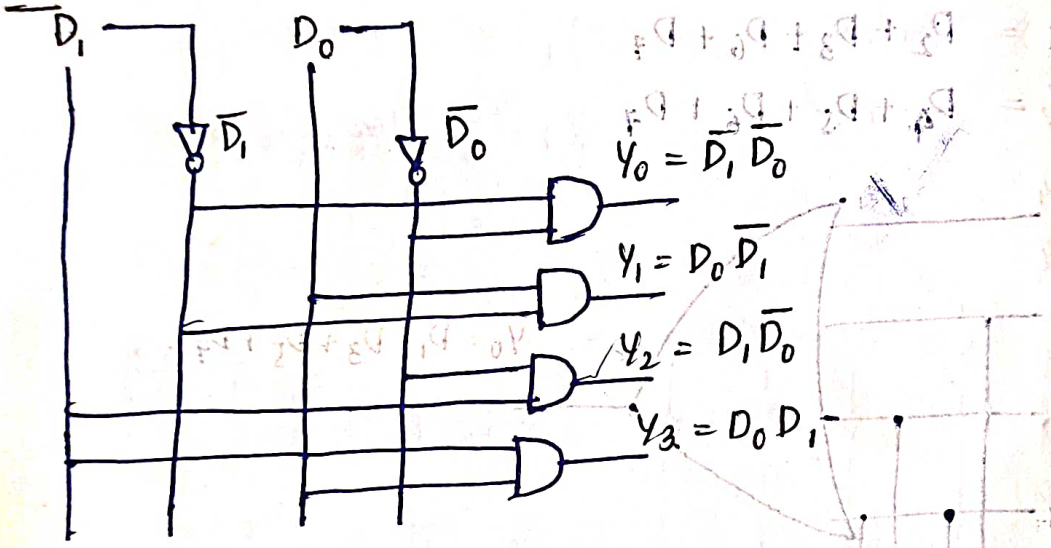
• B.D :



• T.T :

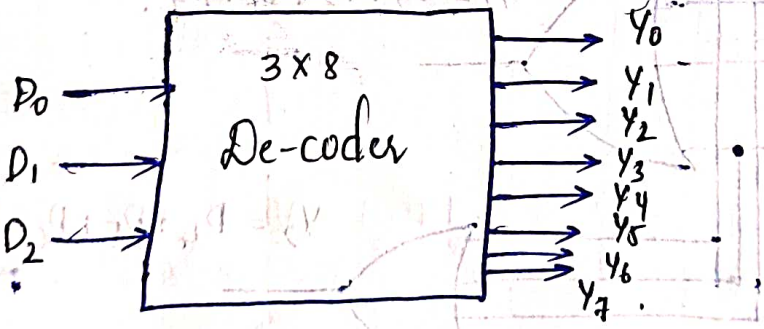
D_1	D_0	Y_3	Y_2	Y_1	Y_0	
0	0	0	0	0	1	: $Y_0 = \bar{D_1}\bar{D_0}$
0	1	0	0	1	0	: $Y_1 = D_0\bar{D_1}$
1	0	0	1	0	0	: $Y_2 = \bar{D_0}D_1$
1	1	1	0	0	0	: $Y_3 = D_1D_0$

• C.D :



* 3x8 :

• B.D :



• T.T :

D_2	D_1	D_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0

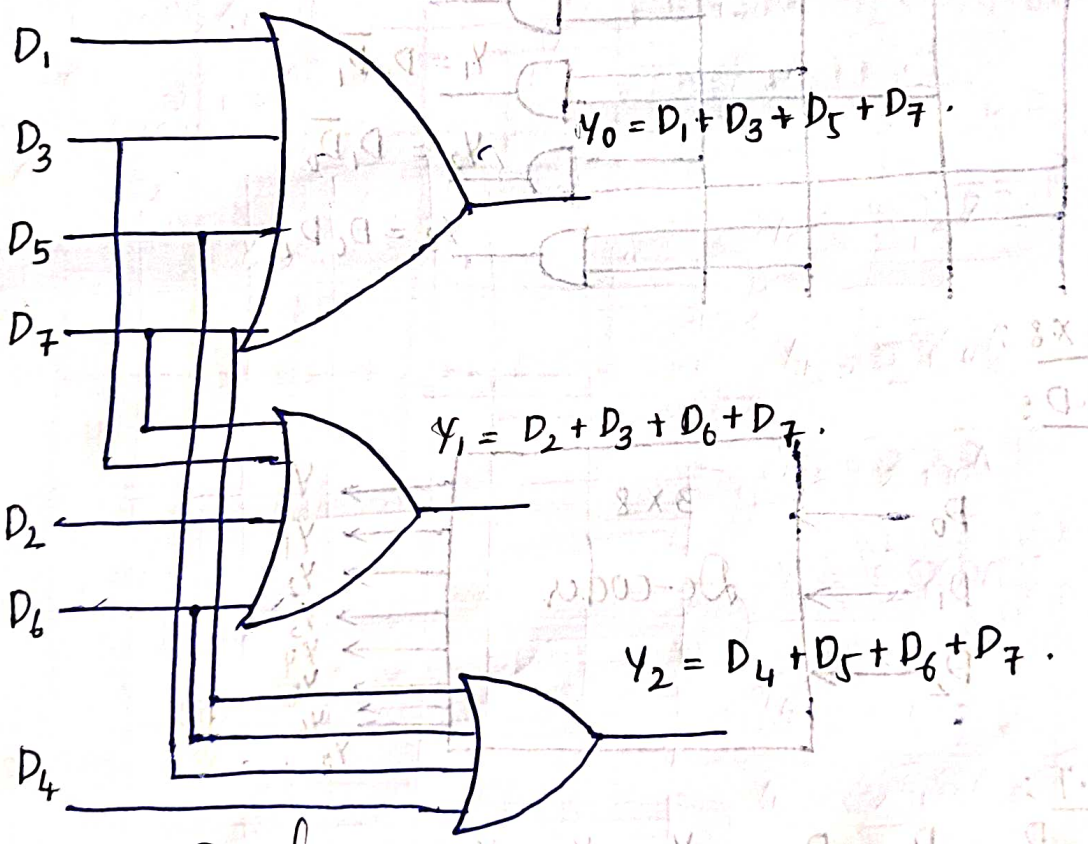
- T.T:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$Y_0 = D_1 + D_3 + D_5 + D_7$.

$Y_1 = D_2 + D_3 + D_6 + D_7$

$Y_2 = D_4 + D_5 + D_6 + D_7$



⇒ Types of Decoders:

1. 2x4
2. 3x8
3. 4x10
4. 4x16

Encoders & De-coders:

⇒ Encoding is a process of encription & de-coding is a process of decryption.

⇒ In encoders, 2^n I/p's - n o/p's

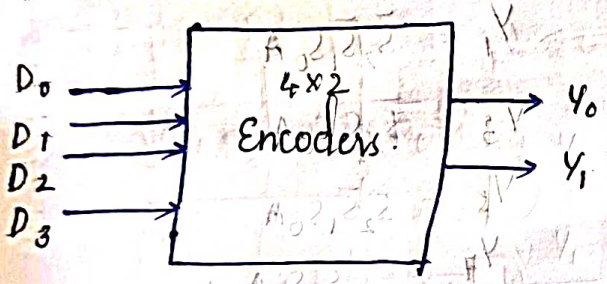
& In decoders, n I/p's - 2^n o/p's.

⇒ Types of Encoders:

1. 4×2
2. 8×3 (octal to binary).
3. 10×4 (decimal to Binary)
4. 16×4

*4×2:

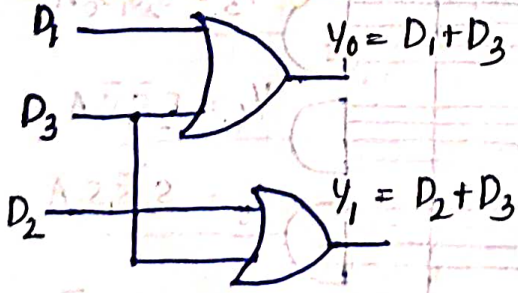
• B.D:



• T.T:

D_3	D_2	D_1	D_0	Y_1	Y_0	
0	0	0	1	0	0	$Y_0 = D_1 + D_3$
0	0	1	0	0	1	$Y_1 = D_2 + D_3$
0	1	0	0	1	0	
1	0	0	0	1	1	

• Circuit Diagram:



* 8×3:

• B.D:

Truth Table:

S_2	S_1	S_0	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
0	0	0	A	0	0	0	0	0	0	0
0	0	1	0	A	0	0	0	0	0	0
0	1	0	0	0	A	0	0	0	0	0
0	1	1	0	0	0	A	0	0	0	0
1	0	0	0	0	0	0	A	0	0	0
1	0	1	0	0	0	0	0	A	0	0
1	1	0	0	0	0	0	0	0	A	0
1	1	1	0	0	0	0	0	0	0	A

$Y_0 = \bar{S}_2\bar{S}_1\bar{S}_0 A$

$Y_1 = \bar{S}_2\bar{S}_1 S_0 A$

$Y_2 = \bar{S}_2 S_1\bar{S}_0 A$

$Y_3 = \bar{S}_2 S_1 S_0 A$

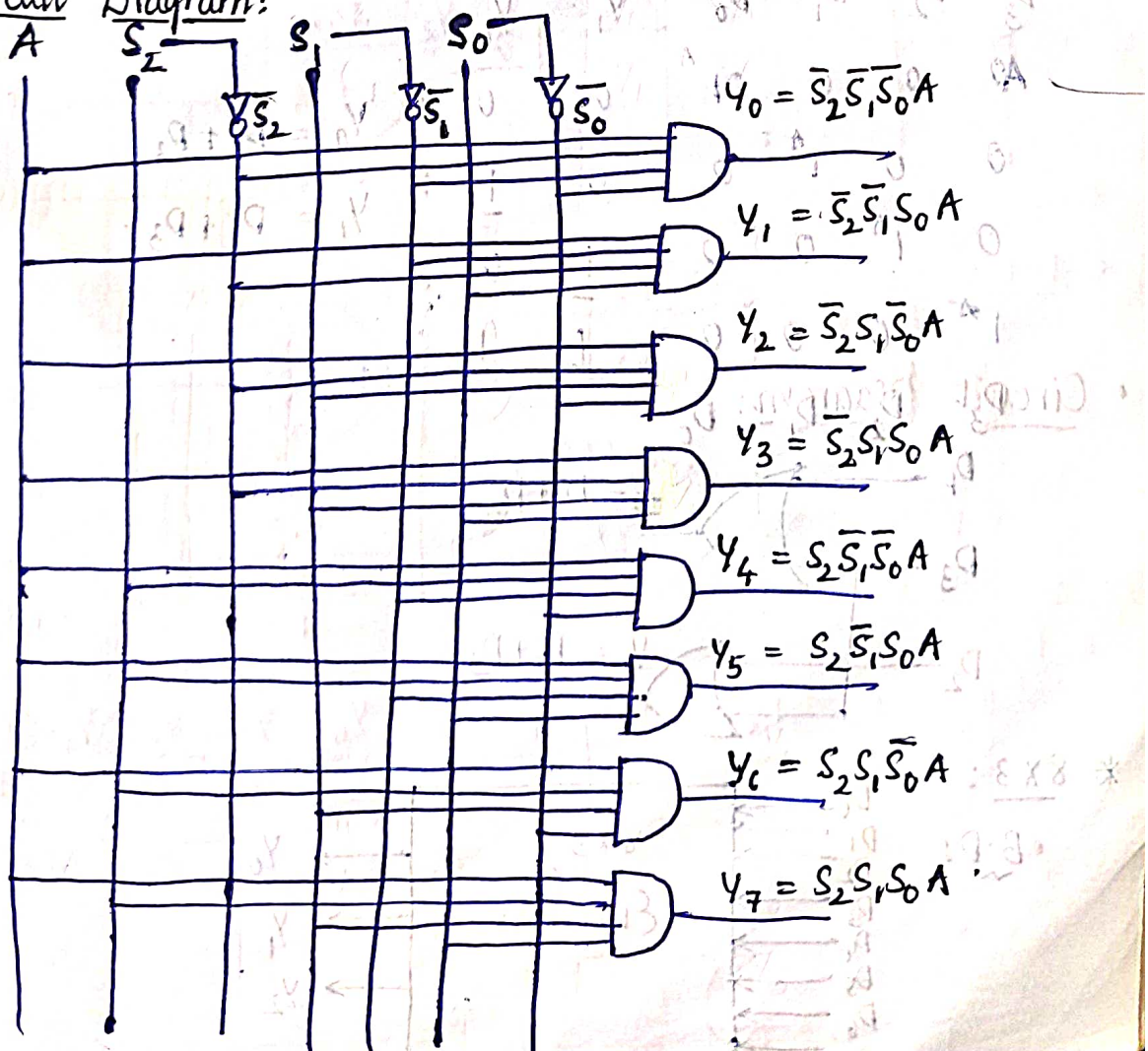
$Y_4 = S_2\bar{S}_1\bar{S}_0 A$

$Y_5 = S_2\bar{S}_1 S_0 A$

$Y_6 = S_2 S_1\bar{S}_0 A$

$Y_7 = S_2 S_1 S_0 A$

Circuit Diagram:



• Truth Table:

S_1	S_0	Y_0	Y_1	Y_2	Y_3
0	0	A	0	0	0
0	1	0	A	0	0
1	0	0	0	A	0
1	1	0	0	0	A

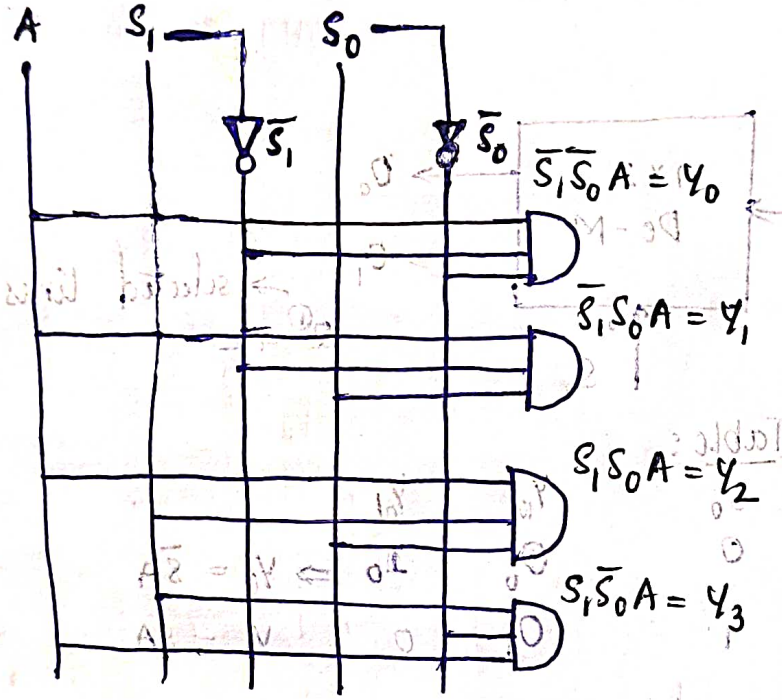
$Y_0 = \bar{S}_1 \bar{S}_0 A$

$Y_1 = \bar{S}_1 S_0 A$

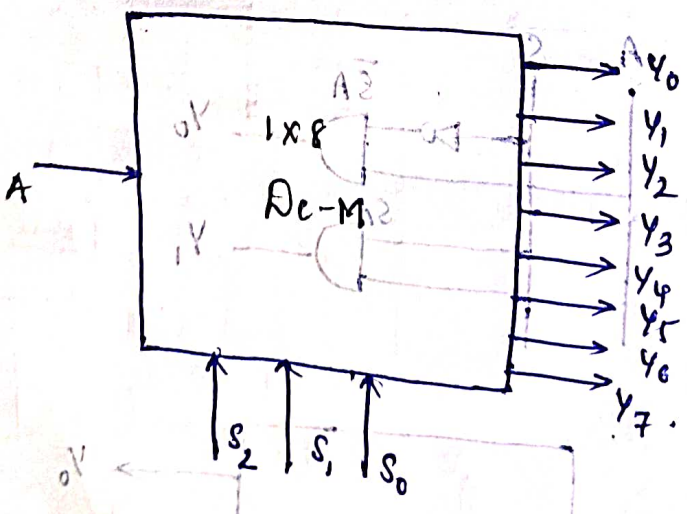
$Y_2 = S_1 S_0 A$

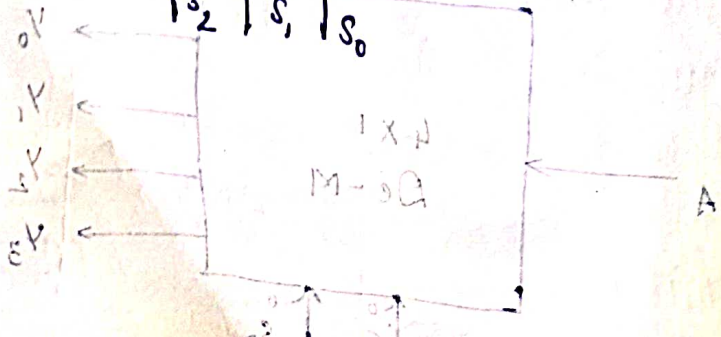
$Y_3 = S_1 \bar{S}_0 A$.

• Circuit Diagram:

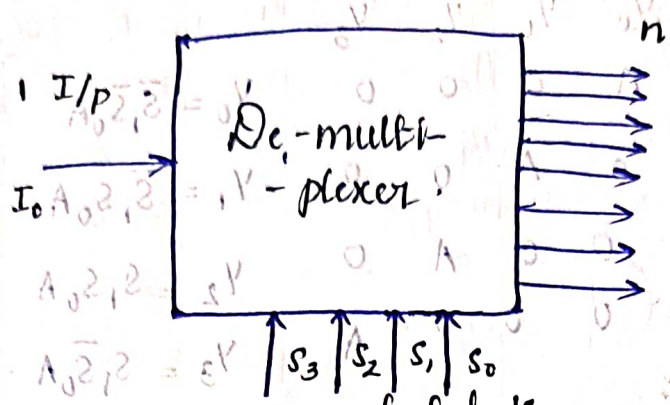


3) 1x8:



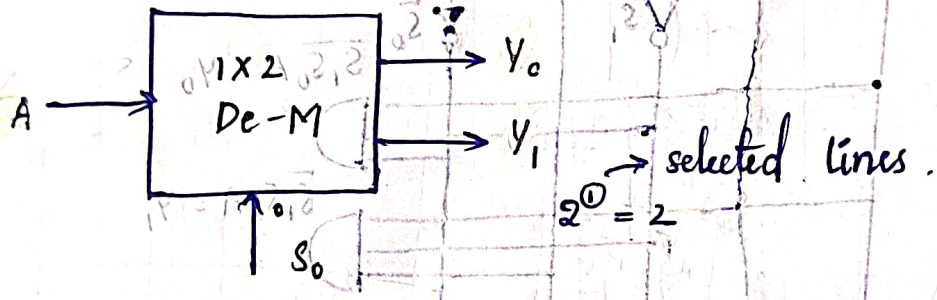


Block Diagram:



Types of Demultiplexers:

1) 1x2



Truth Table:

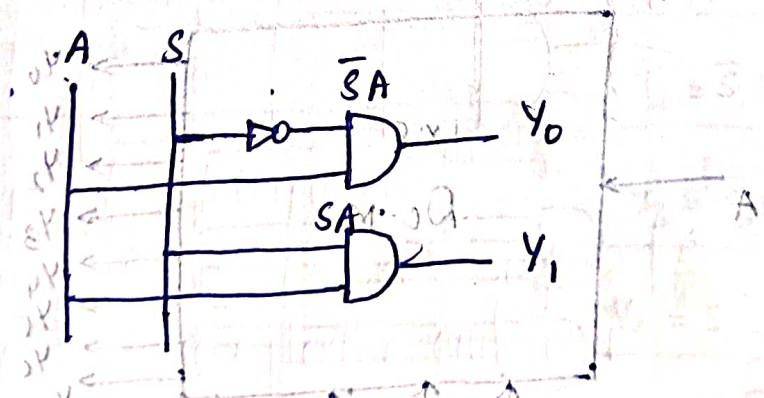
S_0	Y_1	Y_0
0	0	A
1	A	0

$\Rightarrow Y_0 = \bar{S}A$

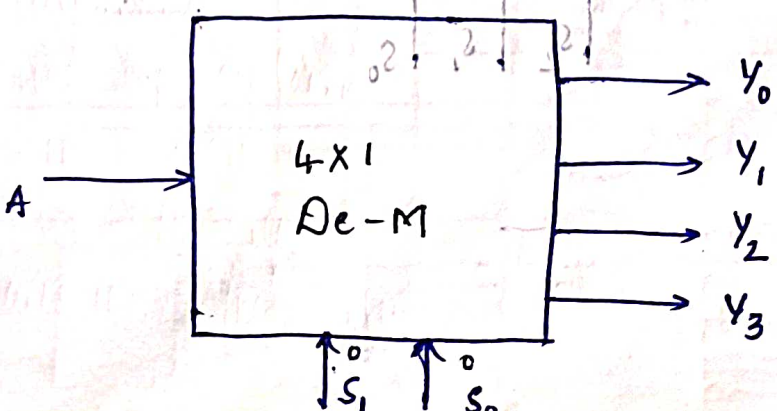
$Y_1 = SA$

Y_1	Y_0
A	0
0	A

Circuit Diagram:



2) 1x4:



⇒ They can be used to implement Integrated Circuits (IC's)

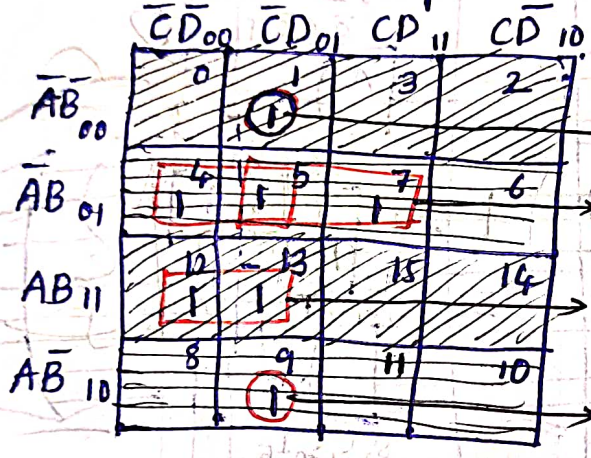
⇒ IC 74153 is a 4×1 multiplexer in which output is same as the input.

⇒ IC 45352 is a 4×1 multiplexer in which the output is complement of the input.

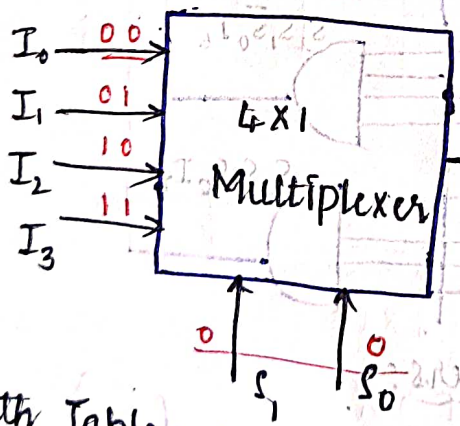
▷ Simplify the given Boolean expression: (using 4x1 multiplexer):

1) $F(A,B,C,D) = \Sigma m(1,4,5,7,9,12,13)$

Step-1: Draw the K-Map.



• B.D:



Ex:

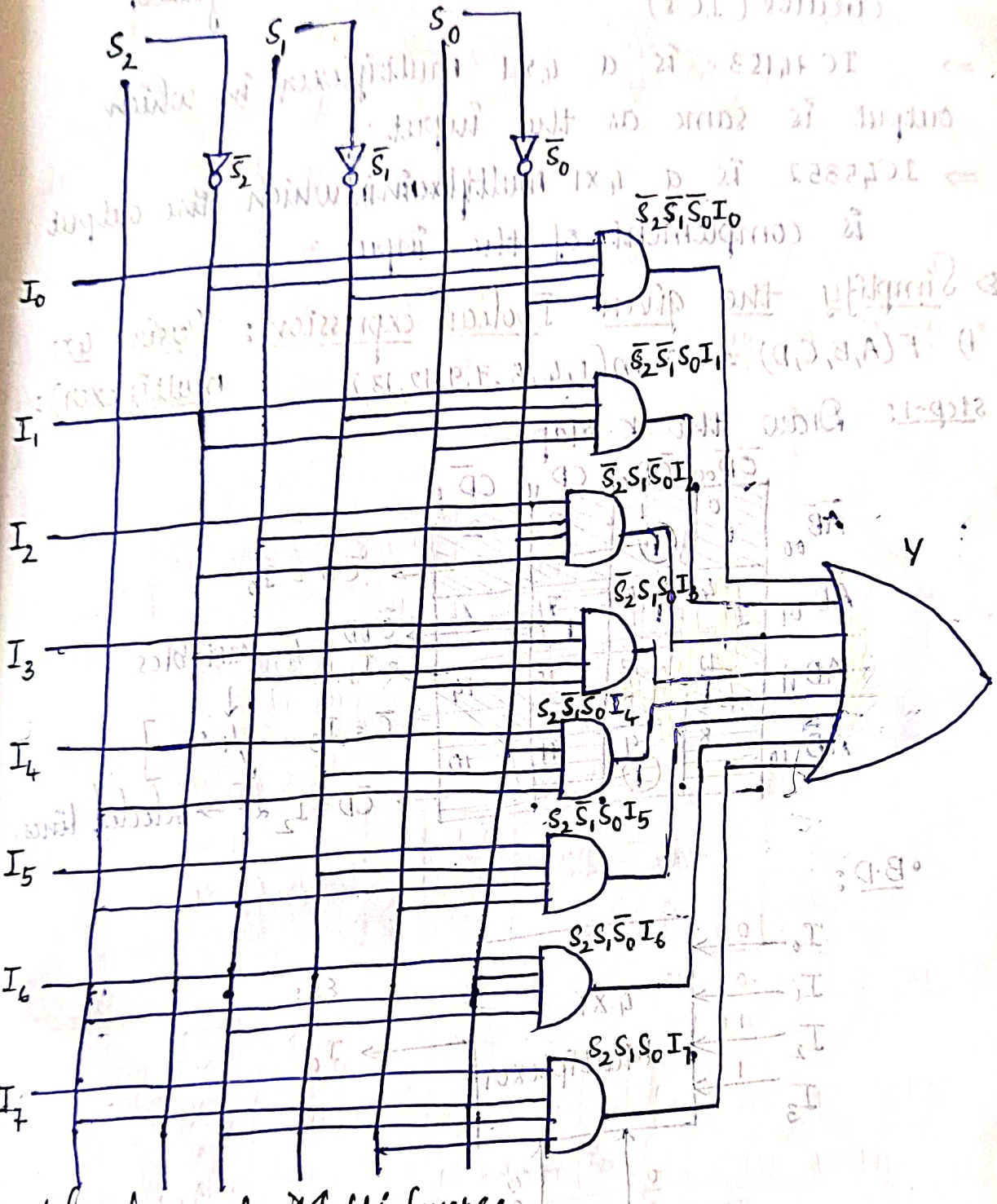
• Truth Table:

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

▷ De-multiplexers:

⇒ It is a combinational circuit with single input and multiple outputs.

Circuit:



⊙ Advantages of Multiplexers:

⇒ It reduces the no. of wires required

⇒ They have less complexity and cost.

⊙ Applications:

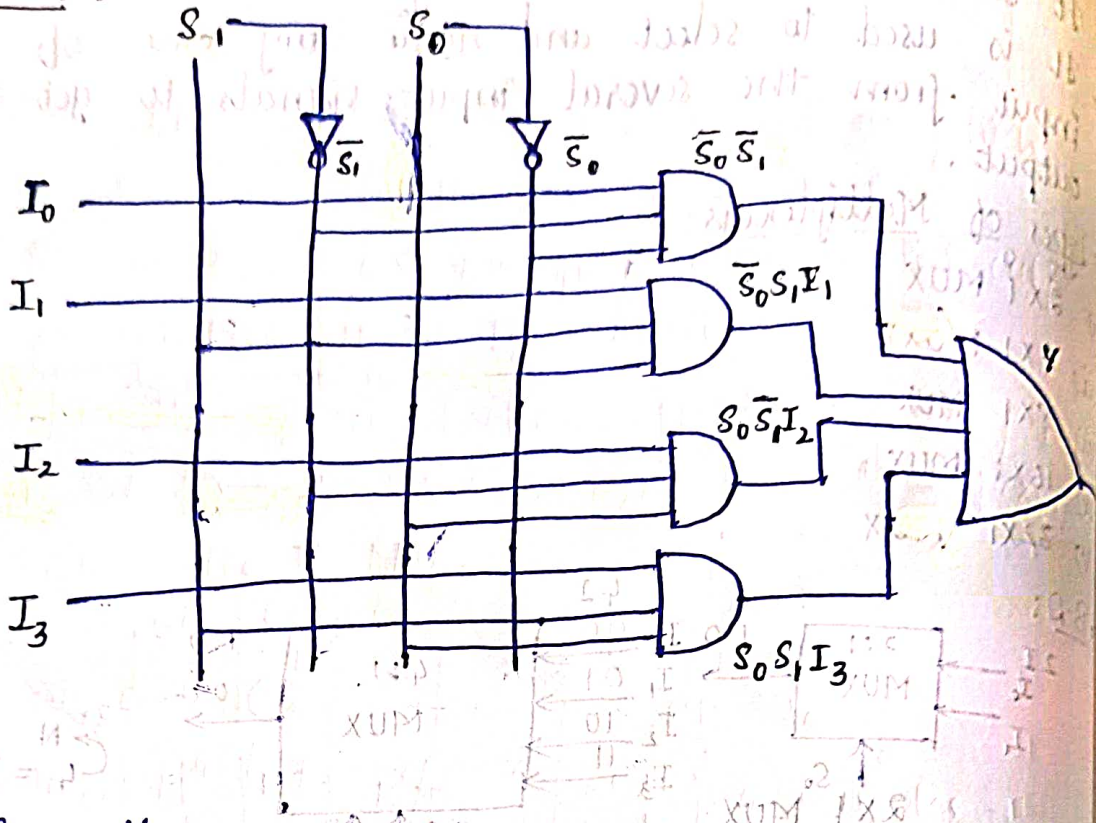
⇒ Communication systems.

⇒ Telephone network.

⇒ Computer Memory.

⇒ Transmissions from the satellite

⊙ Circuit :

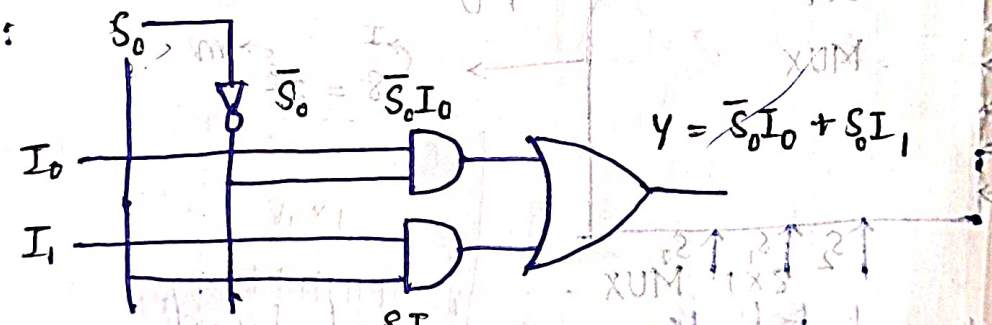


⊙ 2:1

Truth Table :

S_0	Y
0	I_0 $\quad \bar{S}_0 I_0$
1	I_1 $\quad S_0 I_1$

Circuit :



⊙ 8:1 :

Truth Table :

S_2	S_1	S_0	Y	
0	0	0	I_0	$\rightarrow \bar{S}_2\bar{S}_1\bar{S}_0 I_0$
0	0	1	I_1	$\rightarrow \bar{S}_2\bar{S}_1 S_0 I_1$
0	1	0	I_2	$\rightarrow \bar{S}_2 S_1\bar{S}_0 I_2$
0	1	1	I_3	$\rightarrow \bar{S}_2 S_1 S_0 I_3$
1	0	0	I_4	$\rightarrow S_2\bar{S}_1\bar{S}_0 I_4$
1	0	1	I_5	$\rightarrow S_2\bar{S}_1 S_0 I_5$
1	1	0	I_6	$\rightarrow S_2 S_1\bar{S}_0 I_6$
1	1	1	I_7	$\rightarrow S_2 S_1 S_0 I_7$

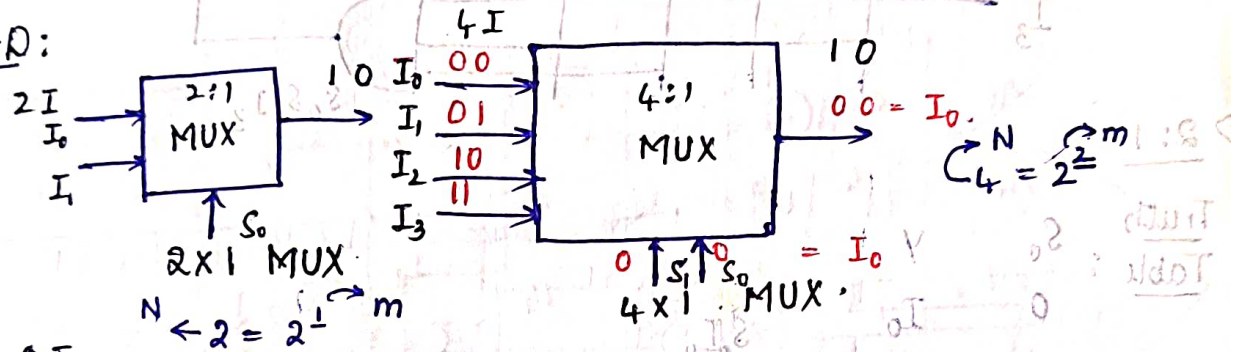
⇒ It can also known as many to one.

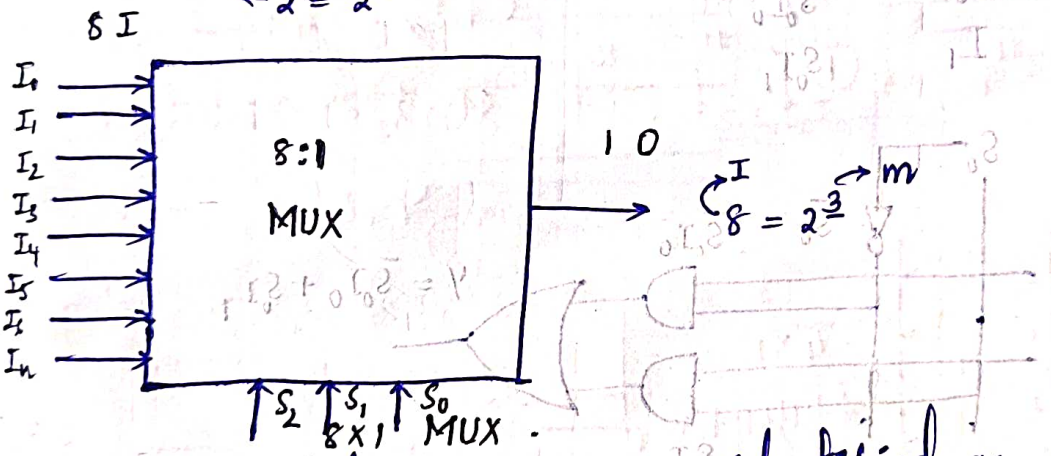
⇒ It is used to select and route any one of the input from the several input signals to get single output.

Types of Multiplexers:

i) 2x1 MUX
ii) 4x1 MUX
iii) 8x1 MUX
iv) 16x1 MUX
v) 32x1 MUX

• B.D:





⇒ No. of selected lines will be identified as $N = 2^m$ where N is the no. of input lines, m is the selected lines

⇒ Depending upon the selected lines, output is produced.

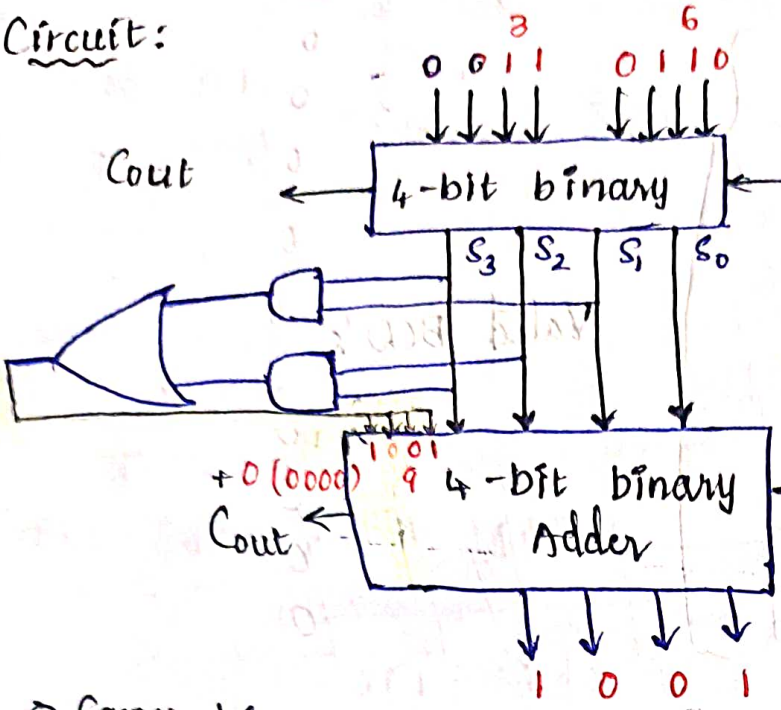
Ex: 4:1

Truth Table:

S_0	S_1	Y	
0	0	I_0	$\rightarrow \bar{S}_0\bar{S}_1I_0$
0	1	I_1	$\rightarrow \bar{S}_0S_1I_1$
1	0	I_2	$\rightarrow S_0\bar{S}_1I_2$
1	1	I_3	$\rightarrow S_0S_1I_3$

⊙ Carry 0:

Circuit:

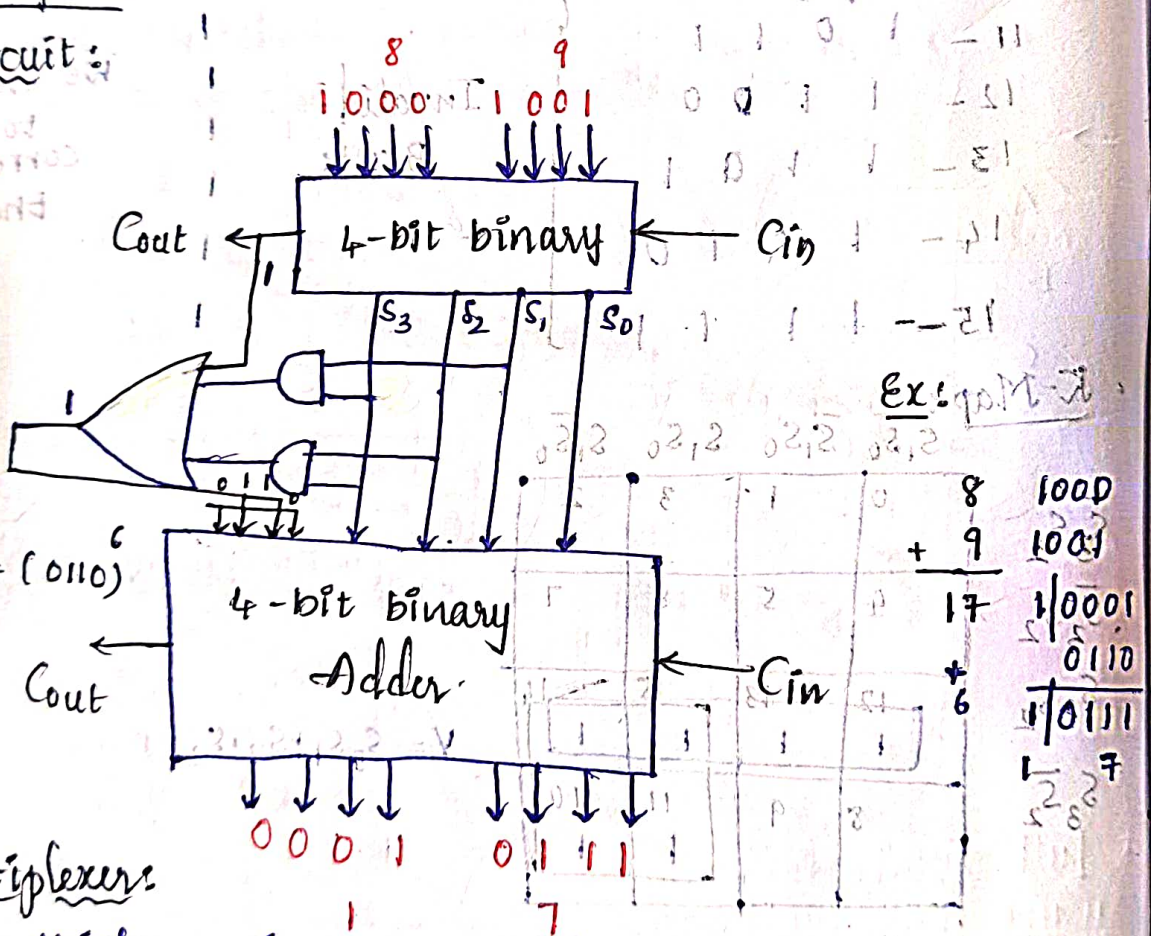


Ex:

3	0011
+6	0110
9	1001

⊙ Carry 1:

Circuit:



Ex:

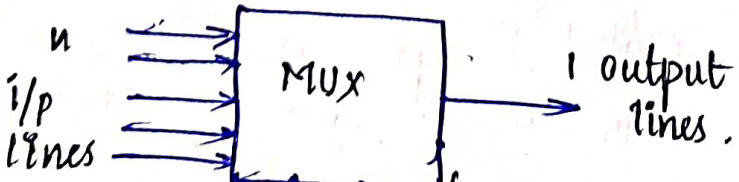
8	1000
+ 9	1001
17	10001
	0110
+ 6	10111
	7

⊙ Multiplexers:

⇒ A multiplexer is a combinational circuit that selects binary info. from one of many input lines and directs it to output line.

⇒ It is also known as MUX.

Circuit:



*

	D S_3	C S_2	B S_1	A S_0	Y
0 –	0	0	0	0	0
1 –	0	0	0	1	0
2 –	0	0	1	0	0
3 –	0	0	1	1	0
4 –	0	1	0	0	0
5 –	0	1	0	1	0
6 –	0	1	1	0	0
7 –	0	1	1	1	0
8 –	1	0	0	0	0
9 –	1	0	0	1	0
10 –	1	0	1	0	1
11 –	1	0	1	1	1
12 –	1	1	0	0	1
13 –	1	1	0	1	1
14 –	1	1	1	0	1
15 –	1	1	1	1	1

Valid BCD's

Invalid BCD's

we need to correct them

• K-Map :

	$\bar{S}_1\bar{S}_0$	$\bar{S}_1 S_0$	$S_1 S_0$	$S_1\bar{S}_0$
$\bar{S}_3\bar{S}_2$	0	1	3	2
$\bar{S}_3 S_2$	4	5	8	7
$S_3 S_2$	12 1	13 1	15 1	14 1
$S_3\bar{S}_2$	8	9	11 1	10 1

$Y = S_3 S_1 + S_3 \cdot S_2$

$S_3 \& S_2$

$S_3 \& S_1$

⊙ BCD Adder: (Binary Coded Decimal).

⇒ The digital system handles the no.'s in the form of BCD.

⇒ A BCD adder circuit adds two BCD digits & produces result also in BCD.

Ex: 526 in BCD.

0101 0010 0110
5 2 6

⊙ Rules:

i) If sum is less than or equal to 9 with carry '0'.

Ex:
$$\begin{array}{r} 3 \\ +\ 6 \\ \hline 9 \end{array} \quad \begin{array}{r} 0011 \\ 0110 \\ \hline 1001 \end{array} \text{ (Valid)}$$

∴ No correction is needed

ii) If sum is greater than 9 with carry '0'.

Ex:
$$\begin{array}{r} 8 \\ +\ 6 \\ \hline 14 \end{array} \quad \begin{array}{r} 1000 \\ 0110 \\ \hline 1110 \end{array} \text{ (Invalid)}$$

∴ Correction is required.

$$+6 \quad 0110$$

$$1\,|\,0100 \text{ (Valid)}$$

(4) match.

iii) If sum is greater than 9 with carry '1'.

Ex:
$$\begin{array}{r} 8 \\ +\ 9 \\ \hline 17 \end{array} \quad \begin{array}{r} 1000 \\ 1001 \\ \hline 0001\,|\,0001 \end{array} \text{ (Invalid)}$$

∴ Correction is required.

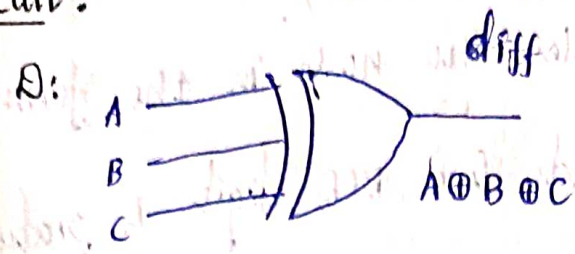
$$+\ 6 \quad 0000\ 0110$$

$$0001\,|\,0111 \text{ (Valid)}$$

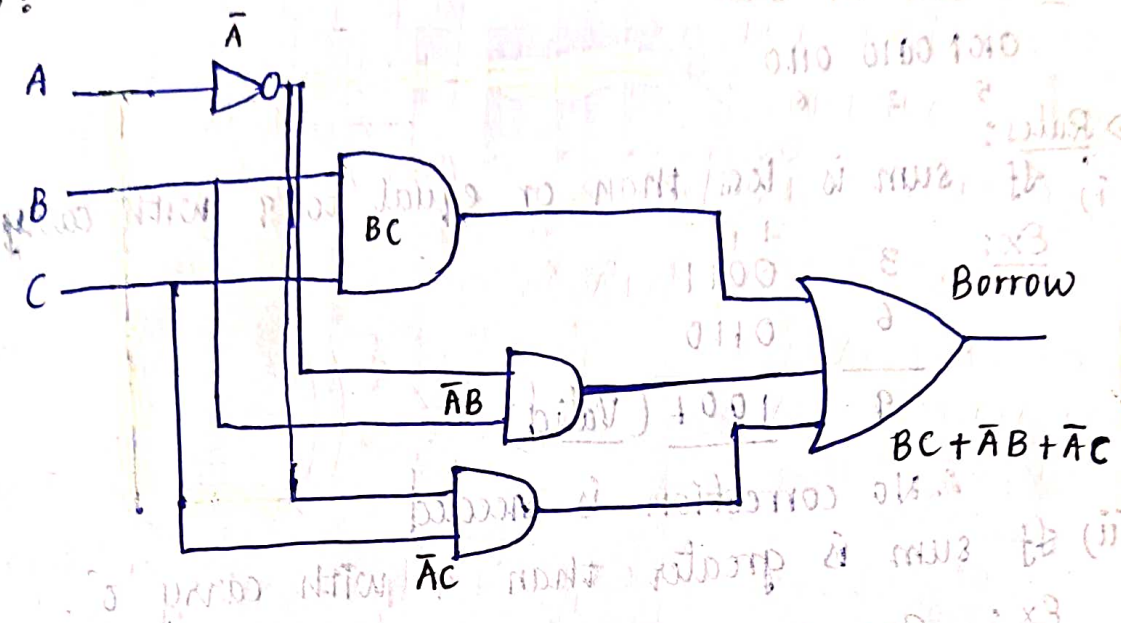
1 7 match

Circuit :

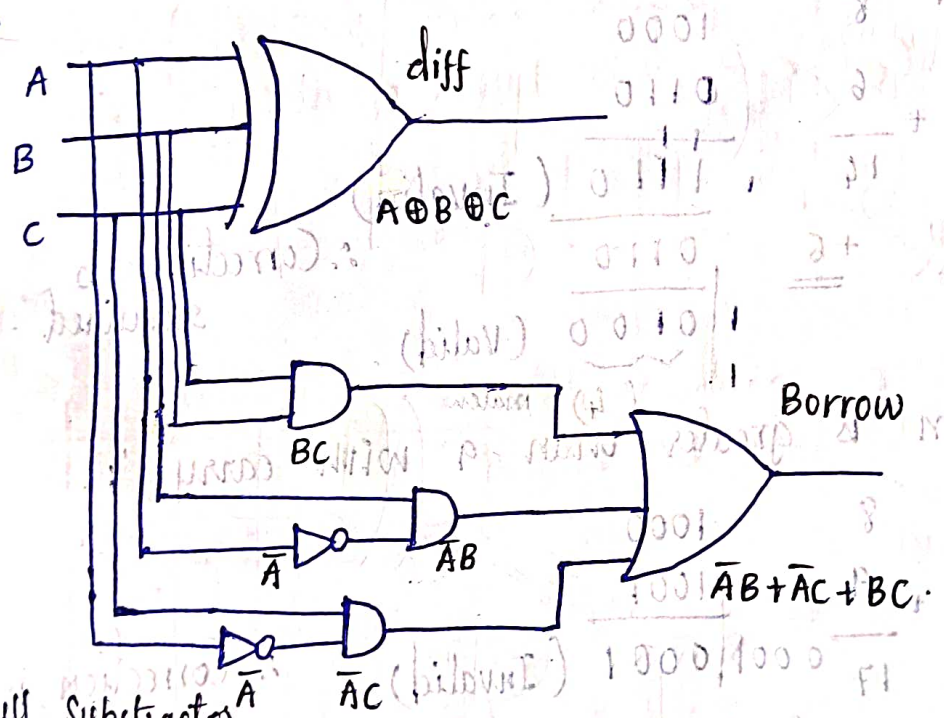
D:



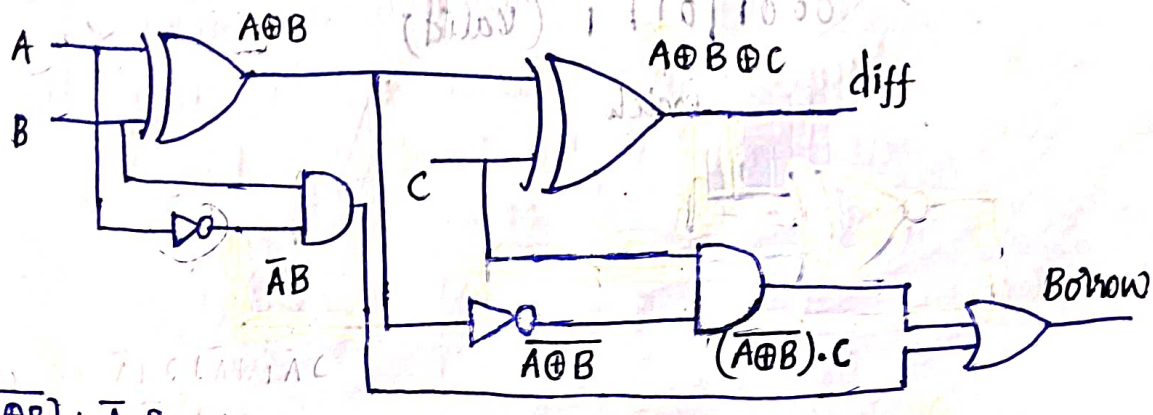
B:



⇒

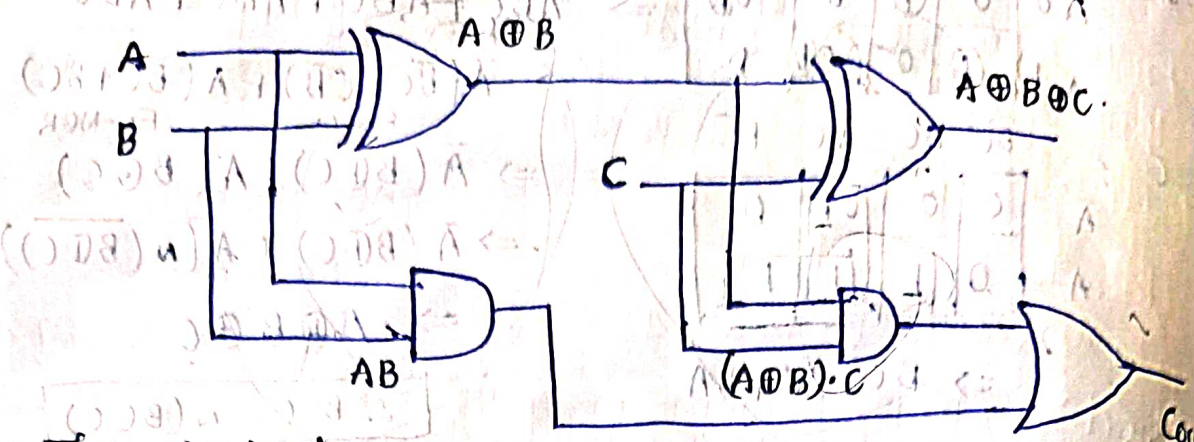


* Full Substractor using half substractor:



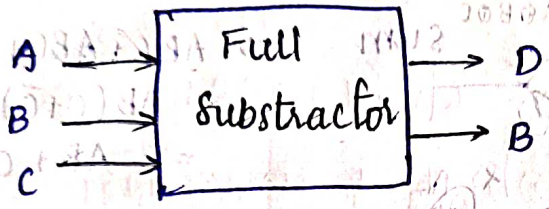
C(A⊕B) + Ā·B

* Full Adder Construction using two half adders:



* Full subtractor:

B·D:



Truth table:

A	B	C	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Maps:

D :

A \ $\bar{B}\bar{C}$	$\bar{B}\bar{C}_{00}$	$\bar{B}C_{01}$	BC_{11}	$B\bar{C}_{10}$
$\bar{A}$ 0	1	0	1	
A 1	1	0	1	0

$\Rightarrow A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}B\bar{C}$

B :

A \ $\bar{B}\bar{C}$	$\bar{B}\bar{C}_{00}$	$\bar{B}C_{01}$	BC_{11}	$B\bar{C}_{10}$
$\bar{A}$ 0	0	1	1	1
A 1	0	0	1	0

$\Rightarrow \bar{A}C + \bar{A}B + BC$

$\Rightarrow A\oplus B\oplus C$

$\Rightarrow \bar{A}C(B+\bar{B}) + \bar{A}B(C+\bar{C}) + BC(A+\bar{A})$

$\Rightarrow \bar{A}CB + \bar{A}C\bar{B} + \bar{A}BC + \bar{A}B\bar{C} + ABC + \bar{A}BC$

$\Rightarrow \bar{A}BC + \bar{A}C\bar{B} + \bar{A}B\bar{C} + ABC \Rightarrow C(AB+\bar{A}\bar{B}) + \bar{A}(C+\bar{C})\cdot B$

K-Maps:

S :

$\bar{A}$	$\bar{B}\bar{C}$ (00)	$\bar{B}C$ (01)	BC (11)	$B\bar{C}$ (10)
$\bar{A}$ (0)	0	1	0	1
A (1)	1	0	1	0

$\Rightarrow \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}B\bar{C}$

$\Rightarrow \bar{A}(\bar{B}C + C\bar{B}) + A(\bar{B}\bar{C} + BC)$ (Ex-OR, Ex-NOR)

$\Rightarrow \bar{A}(B \oplus C) + A(B \odot C)$

$\Rightarrow \bar{A}(B \oplus C) + A(\overline{B \oplus C})$

$\Rightarrow A \oplus B \oplus C$

$\therefore B \odot C = \overline{(B \oplus C)}$

C :

A	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0	0	1	0
A	0	1	1	1

$\Rightarrow BC + AC + BA$

$\Rightarrow BC(A+\bar{A}) + AC(B+\bar{B}) + AB(C+\bar{C})$

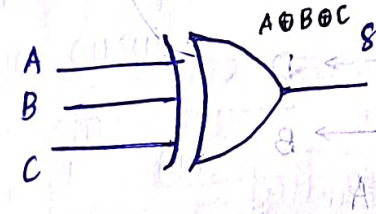
$\Rightarrow ABC + \bar{A}BC + A\bar{B}C + AB\bar{C}$

$\Rightarrow AB(C+\bar{C}) + C(\bar{A}B + A\bar{B})$

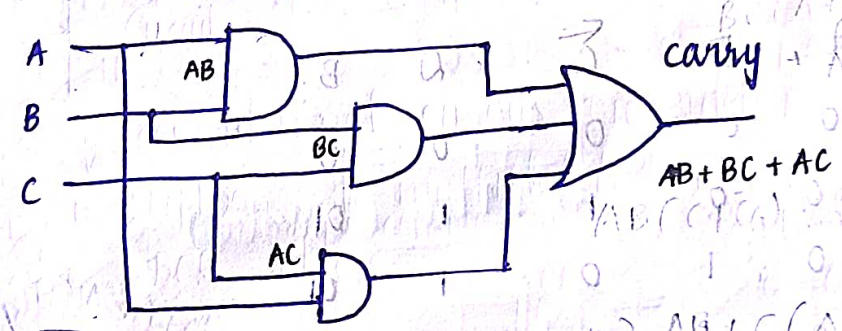
$\Rightarrow AB + C(A \oplus B)$

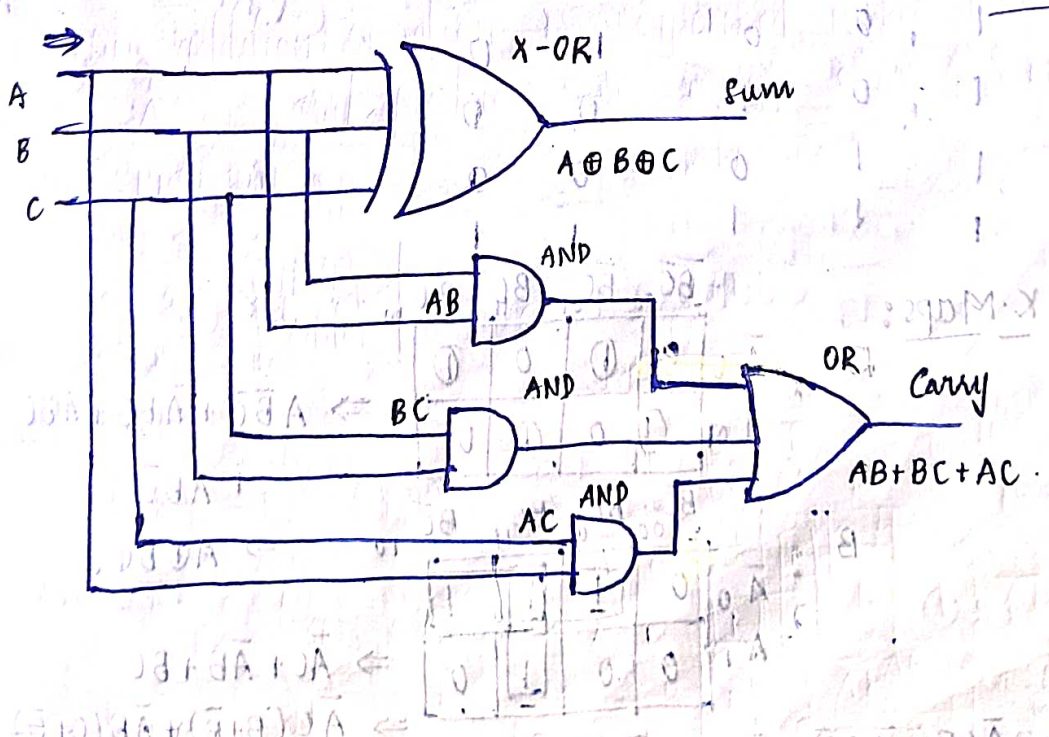
Circuit:

Sum:

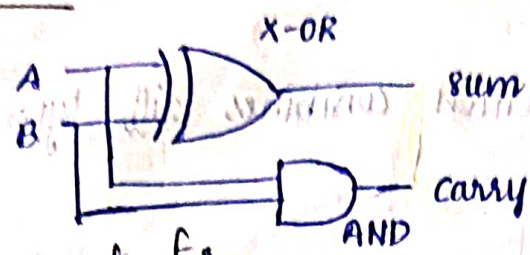


carry:



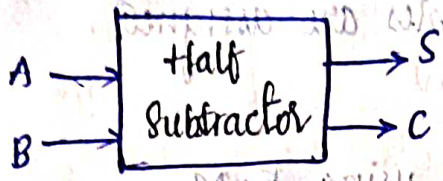


⇒ Circuit:



* Half Subtractor:

B.D:



Truth Table:

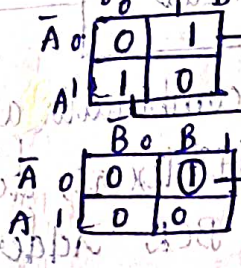
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Binary (2) 0 Borrow 1 −1

S → ① → $\bar{A}B$ (NOT)

C → $\bar{A}B + \bar{B}A \equiv A \oplus B$ (XOR)

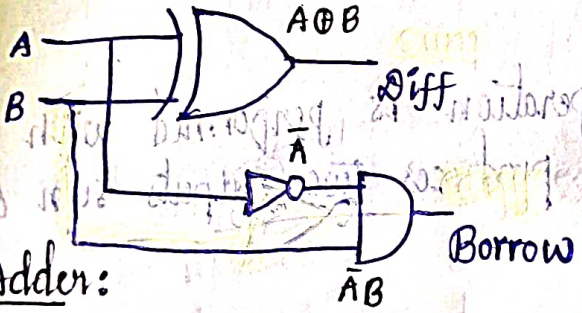
K Maps:



$\Rightarrow \bar{A}B + \bar{B}A = A \oplus B$ (XOR)

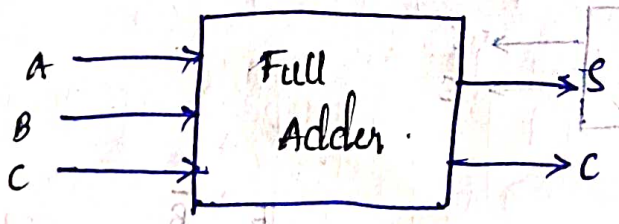
(NOT)

Circuit:



* Full Adder:

B.D:



Truth Table:

A	B	C	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

* 2. Combinational Circuits *

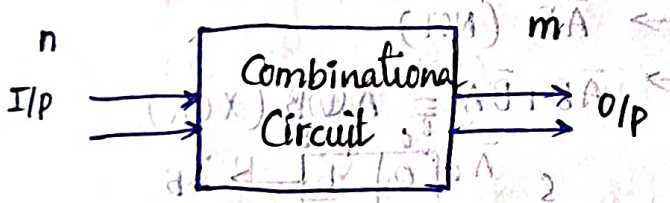
Combinational Circuits:

- They are the circuits which combines diff types of logic gates.

⇒ How to Design?

- Problem to be stated.
- Input & output variables are assigned.
- Construct truth tables.
- Simplify boolean funcs. using K-Maps.
- Draw logic circuit.

⇒ Block Diagram:



⇒ Diff. types of combinational circuits are adders, subtracters, multiplexers, de-multiplexers, encoders & decoders, BCD Adders.

* Half-Adder:

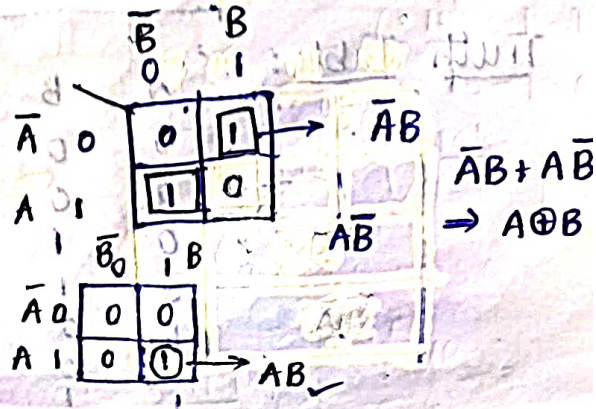
⇒ In this, addition operation is performed with two inputs A & B and produces two outputs sum & carry.

⇒ Block Diagram:

⇒ Truth Table:

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

S:



C:

r's : $(16)_{10}^{2} - (F9)_{16}$

$\Rightarrow (256)_{10} - (249)_{10}$

$\Rightarrow (7)_{10}$

$\Rightarrow (7)_{16}$.

* Note : For r's complement, we can apply only NOT gate (—▷○—).

⇒ R's complement:

$\{(r^n)_{10} - N\} \rightarrow$

where, r → radix / base
n → no. of digits in the data
N → Given no. / data.

⇒ (r-1)'s complement:

$\{((r^n)_{10} - 1) - N\}$

Eg:

* $(172)_8$ =

Sol: n=3, r=8, N=172

(r-1)'s ⇒ $((8)^3_{10} - 1) - (172)_8$

⇒ $(512-1)_{10} - (122)_{10}$

⇒ $(511-122)_{10}$

⇒ $(389)_{10}$

⇒ $(605)_8$

* 7's complement

n=3 : 777 − 172 = 605

* D to O: 8 | 389; 8 | 48 − 5; 6 − 0

r's:

$((8)^3_{10}) - (172)_8$

⇒ $(512)_{10} - (172)_8$

⇒ $(512)_{10} - (122)_{10}$

⇒ $(390)_{10}$

⇒ $(606)_8$.

* D to O: 8 | 390; 8 | 48 − 6; 6 − 0

* 8's C: 777+1 − 172 = 606

* O to D: 172 → $8^2\ 8^1\ 8^0$

⇒ 2 + 56 + 64

⇒ $(122)_{10}$

3 64 8 512

* $(F9)_{16}$:

r-1's: $[(16)^2_{10} - 1] - (F9)_{16}$

⇒ $(256-1)_{10} - (249)_{10}$

⇒ $(255)_{10} - (249)_{10}$

⇒ $(6)_{10}$

⇒ $(6)_{16}$

* 15's: F-9; 15−9=6

* 16's: 15+1−9 = 7

* H.D to D: F − 15 9 → $16^1\ 16^0$

⇒ 9 + 240

⇒ $(249)_{10}$

* D to H.D $(6)_{10} \rightarrow (6)_{16}$ 16 | 16 | 0-6

complement:

* Eg: 1) 110 → 6
(+ve no.)

⇒ 0110

(r-1)
⇒ 1's complement (Toggle) 1001 (-ve no.)

r
⇒ 2's complement : 1001 + 1 = 1010 (Result)

(OR gate)

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

1 1 1 x 0 carry 1

2) $(127)_{10}$

Sol:

127 →	2 | 127
(r-1) 1's: 1111111	2 | 63 - 1
0000000	2 | 31 - 1
r 2's: + 1	2 | 15 - 1
0000001 (Result)	2 | 7 - 1
	2 | 3 - 1
	1 - 1

3) $(155)_{10}$

Sol: (r-1)'s : r = 10, n = 3, N = 155

$\Rightarrow [(10)^3_{10} - 1] - (155)_{10}$

$\Rightarrow (999)_{10} - (155)_{10}$

$\Rightarrow \underline{(844)_{10}}$

r's : $[(10)^3_{10}] - (155)_{10}$

$\Rightarrow (1000)_{10} - (155)_{10}$

$\Rightarrow \underline{(845)_{10}}$

Cross check:

No. of digits : 3

• 9's complement : 999 − 155 = 844

• 10's complement :

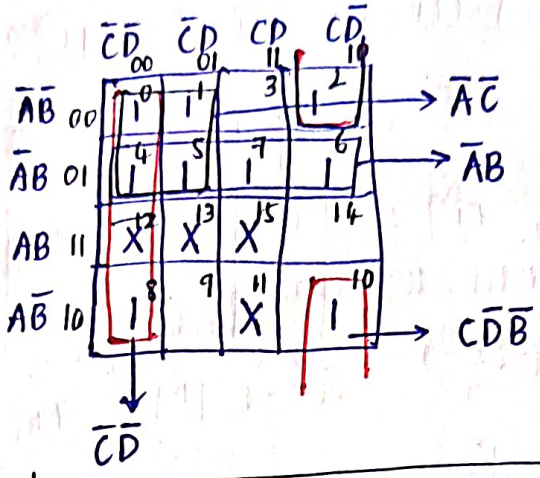
1 ← 999 + 1

0 ← −155

845

* $F = m(0,1,2,4,5,6,7,8,10) + d(11,12,13,15)$.

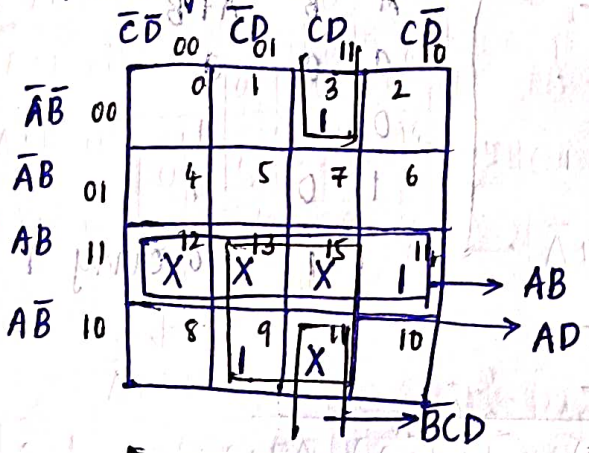
Sol:



$\therefore F = \bar{A}\bar{C} + \bar{A}B + \bar{C}\bar{D} + \bar{B}C\bar{D}$

* cross check:

Missing terms : (3,9,14)



$\therefore F = AB + AD + \bar{B}CD$

◇ Number System:

⇒ We can represent no.'s in the form of 0's & 1's. in computer system.

⇒ We can also represent no.'s in signed & unsigned manners & to perform arithmetic operations

⇒ To represent -ve no.'s, we use complements.

⇒ Sign bit 0 represents +ve, 1 represents -ve no.

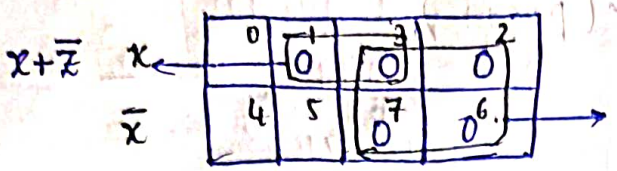
◇ complement (Invert / Toggle).

0 → 1

1 → 0

⇒ For every conversion, we can apply r & $(r-1)$'s complement.

* $F(x,y,z) = \Pi(1,2,3,6,7)$.
$y+z \quad y+\bar{z} \quad \bar{y}+\bar{z} \quad \bar{y}+z$



$\therefore F = (x+\bar{z}).\bar{y}$

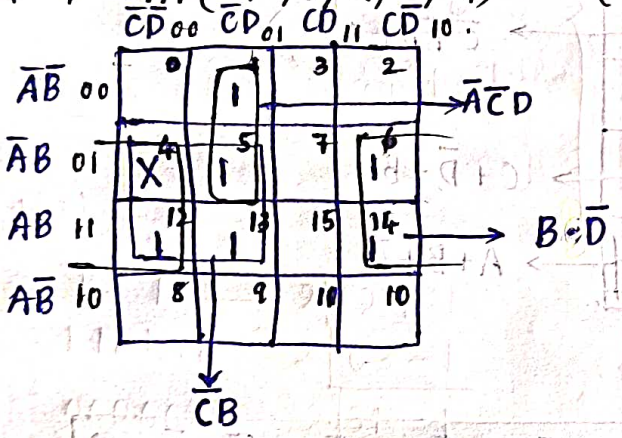
⊙ Don't Care Condition: (X) — Cross

⇒ It says that we can make largest group of cells.

⇒ In grouping, we can only consider don't care condition when it is needed, in rimng positions it will be ignored.

Eg: * $F = m(1,5,6,12,13,14) + d(4)$.

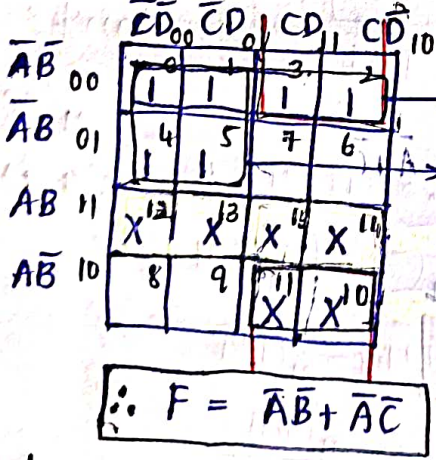
Sol:



$\therefore F = B\bar{C} + B\bar{D} + \bar{A}\bar{C}D$

* $F = m(0,1,2,3,4,5) + d(10,12,11,13,14,15)$.

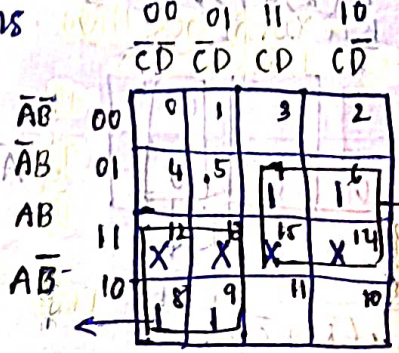
Sol:



no need to consider

$\therefore F = \bar{A}\bar{B} + \bar{A}\bar{C}$

* Cross check: Missing terms ⇒ (6,7,8,9)

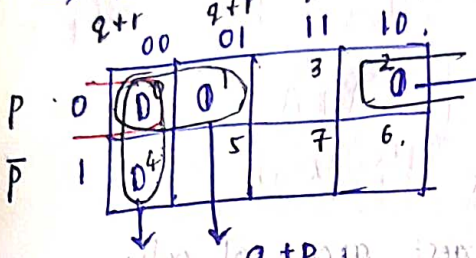


$\therefore F = A\bar{C} + BC$

Canonical POS to POS :

* $F=(P+Q+R)\cdot(\bar{P}+Q+R)\cdot(P+\bar{Q}+R)\cdot(P+Q+\bar{R})$.

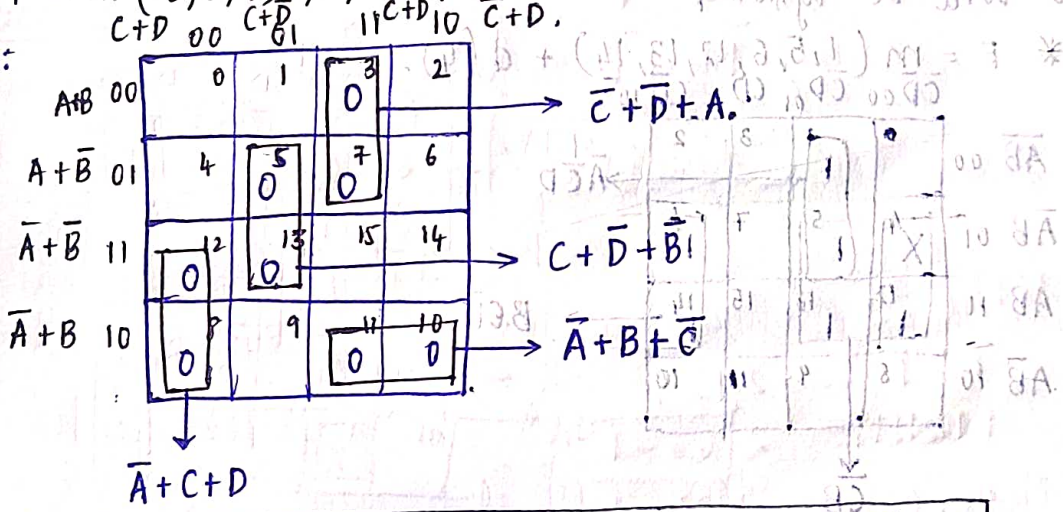
Sol: $\Rightarrow 2^3 = 8$



$\therefore F = (P+Q)\cdot(Q+R)\cdot(P+R)$

* $F = \Pi^{(A,B,C,D)}(3,5,7,8,10,11,12,13)$

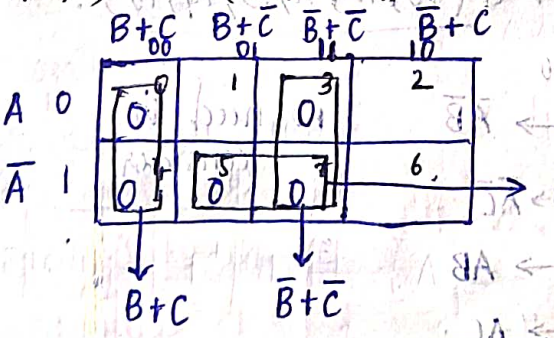
Sol:



$\therefore F = (\bar{A}+C+D)\cdot(C+\bar{D}+\bar{B})\cdot(\bar{A}+B+\bar{C})\cdot(\bar{C}+\bar{D}+A)$

* $F(A,B,C) = \Pi(3,4,5,0,7)$

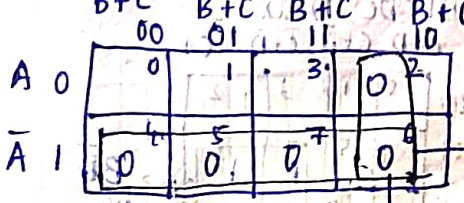
Sol:



$\therefore F = (B+C)\cdot(\bar{B}+\bar{C})\cdot(\bar{A}+\bar{C})$

* $F(A,B,C) = \Pi(2,4,5,6,7)$

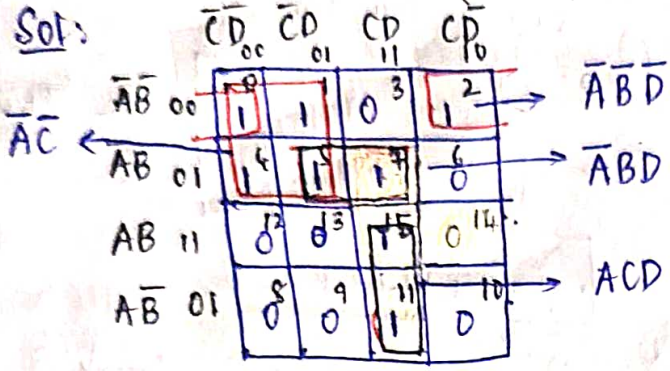
Sol:



$\therefore F = \bar{A}\cdot(\bar{B}+C)$

* $F(A,B,C,D) = \Sigma(0,1,2,4,5,7,11,15)$

Sol:

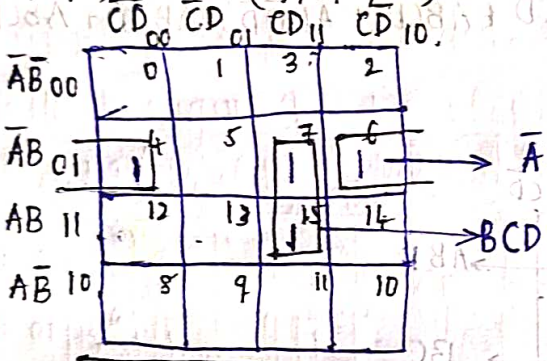


5 or 7 (or) 7 & 15 consider only one

$\therefore F = \bar{A}\bar{C} + \bar{A}\bar{B}\bar{D} + \bar{A}BD + ACD.$

* $F(A,B,C,D) = \Sigma(4,6,7,15)$

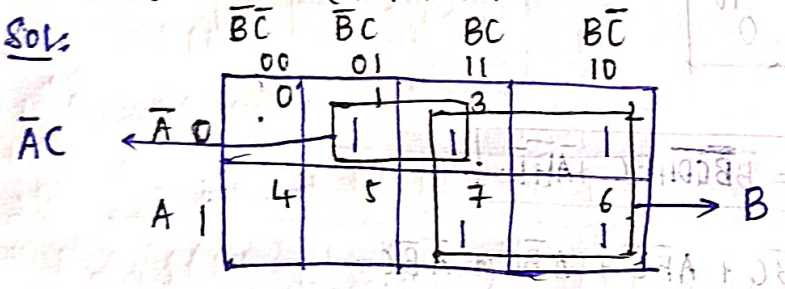
Sol:



$\therefore F = BCD + \bar{A}B\bar{D}$

* $F(x,y,z) = \Sigma(1,2,3,6,7).$

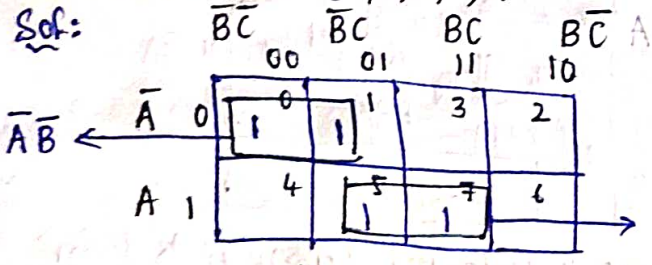
Sol:



$\therefore F = B + \bar{A}C.$

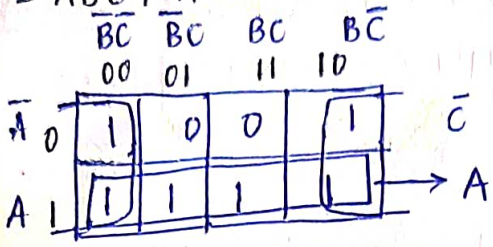
* $F(A,B,C) = \Sigma(0,1,5,7).$

Sol:



$\therefore F = \bar{A}\bar{B} + AC.$

* $Y = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}BC + ABC + A\bar{B}C + AB\bar{C}$. (SOP)



$2^3 = 8$

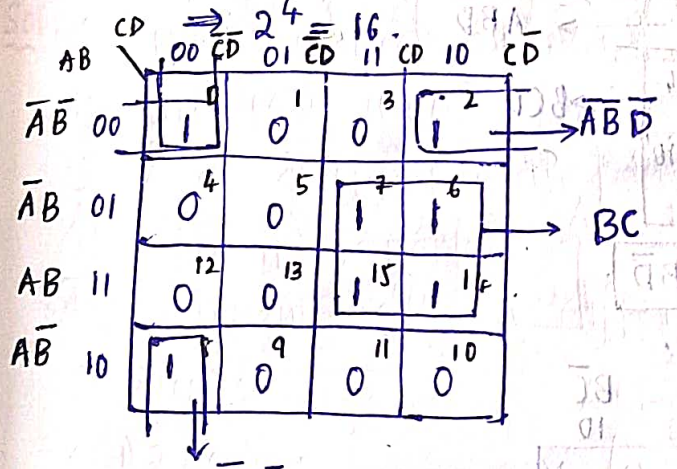
$A + \bar{A}\bar{C}$
$\Rightarrow A(B+\bar{B})(C+\bar{C}) + \bar{A}\bar{C}(B+\bar{B})$

$\therefore \boxed{F = A + \bar{C}}$

* Note: [For 5 variables, 2 – 16 K-Maps are used]
For $E=0$ & For $E=1$

* $Y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + ABCD$
(2^4 2^4)

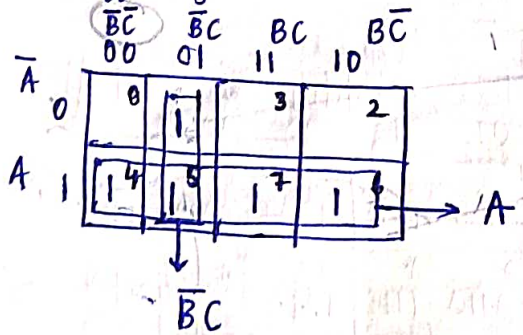
Sol: For 4 variables,



$\therefore Y = \bar{B}\bar{C}\bar{D} + BC + \bar{A}\bar{B}\bar{D}$.

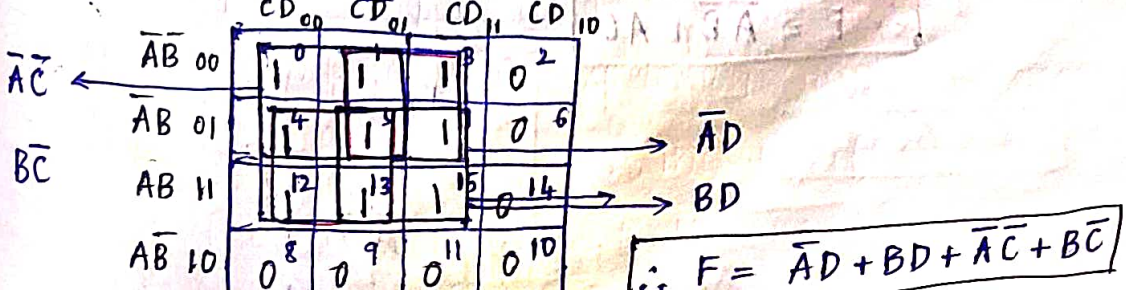
* $F = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$.

$\Rightarrow 2^3 = 8$



$\therefore F = A + \bar{B}C$

* $F(A,B,C,D) = \sum m\,(0,1,3,4,5,7,12,13,15)$ (SOP)



$\therefore F = \bar{A}D + BD + \bar{A}\bar{C} + B\bar{C}$

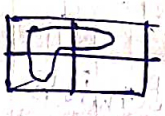
⇒ Fill cells of K-map for SOP with 1. respective to the min terms.

⇒ Fill cells of K-map for POS with 0 w.r.t the max term

⇒ Create rectangular groups that contains total terms in the power of 2 i.e, 2^n.

With the help of these groups we find the simplified expression.

⇒ Notice that each group should have largest no. of 1's for min terms & 0's for max terms. A group can't contain an empty cell or an cell with 0 → SOP, POS → 1.

[Note:  we can map 1,2,4 cells → 2^n but we can't map 3 cells.]

Q. * $F = \bar{A}\bar{B} + \bar{A}B + AB$. (SOP)

⇒

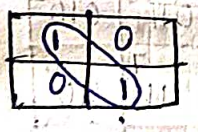
	$\bar{B}$ 0	B 1
$\bar{A}$ 0	1	1
A 1	0	1

→ $\bar{A}$ [2 variables : ⇒ $2^2 = 4$]

↓ B

∴ $F = \bar{A} + B$

• Rules:

① 

②

1	0	0	1
1	1	1	1
1	1	1	1
1	0	0	1

Largest no. of 1's → 1 group.

8 → 1 group ✓

4 → 3 groups ✗

Q. * $F = pqr + \bar{p}qr + p\bar{q}r + \bar{p}\bar{q}r$. (POP)

$2^3 = 8$

	qr 00	$\bar{q}r$ 01	$q\bar{r}$ 11	$\bar{q}\bar{r}$ 10
$\bar{P}$ 0	0	0	1	0
P 1	0	1	1	1

↓ pr ↓ qr → pq

∴ $F = pr + qr + pq$.

$pr(q+q')$
$+ qr(p+p')$
$+ pq(r+r')$ } 6

[$pqr + pqr + pqr = pqr$].

POS to SOP:

* $F = M[(x+y+z)\cdot(x+\bar{y}+z)\cdot(\bar{x}+y+z)\cdot(\bar{x}+\bar{y}+z)]$

Sol: POS : 0 2 4 6

Missing terms : 1 3 5 7

$\therefore \sigma(m_{1}+m_{3}+m_{5}+m_{7})$

$001 \quad 011 \quad 101 \quad 111$

SOP :

$\sigma(\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz)$

$\therefore F = \sigma(\bar{x}\bar{y}z + \bar{x}yz + x\bar{y}z + xyz)$

Karnaugh Maps (K-Maps):

⇒ It is introduced to simplify boolean expressions in an easy way.

⇒ It is a graphical method which consists of 2^n cells for n variables.

⇒ The adjacent cells are differed only in single bit position.

Ex: 1) 2 cells:

$\Rightarrow 2^2 = 4$

	$\bar{B}_0$	B_1
$\bar{A}$ 0	00 (0)	01 (1)
A 1	10 (2)	11 (3)

2) For 3 cells: (3 variable)

$\Rightarrow 2^3 = 8$

	$\bar{B}$ 00	B 01	$\bar{C}$ 11	C 10
$\bar{A}$ 0	000 (0)	001 (1)	011 (3)	010 (2)
A 1	100 (4)	101 (5)	111 (7)	110 (6)

3) For 4 variables:

AB \ CD	00	01	11	10
00	0000 (0)	0001 (1)	0011 (3)	0010 (2)
01	0100 (4)	0101 (5)	0111 (7)	0110 (6)
11	1100 (12)	1101 (13)	1111 (15)	1110 (14)
10	1000 (8)	1001 (9)	1011 (11)	1010 (10)

Steps to simplify the expressions using K-maps:

1. Find the K-map as per the no. of variables.
2. Find the min & max terms in the given

b) $F = (\bar{A}+B)(C+\bar{B})$

Sol: $(\bar{A}+B+C\cdot C')\cdot(\bar{B}+C+A\cdot A')$

$\Rightarrow (A'+B+C)\cdot(A'+B+C')\cdot(A+B'+C)\cdot(A'+B'+C)$.

A	B	C	Term	Max Term
0	0	0	A+B+C	m_0
0	0	1	A+B+C'	m_1
0	1	0	A+B'+C	m_2
0	1	1	A+B'+C'	m_3
1	0	0	A'+B+C	m_4
1	0	1	A'+B+C'	m_5
1	1	0	A'+B'+C	m_6
1	1	1	A'+B'+C'	m_7

∴ Short hand notation:

$\pi(m_4, m_5, m_2, m_6)$

$\Rightarrow \pi(2,4,5,6)$.

c) $(x+y)\cdot(x+z)$

Sol: $(x+y+z\cdot z')\cdot(x'+z+y\cdot y')$

$\Rightarrow (x+y+z)\cdot(x+y+z')\cdot(x'+y+z)\cdot(x'+y'+z)$

∴ Short hand notation:

$\Rightarrow \pi(m_4, m_1, m_0, m_6)$

$\Rightarrow \pi(0,1,4,6)$

d) SOP to POS:

* $F = m[\bar{x}yz + x\bar{y}z + xyz + \bar{x}\bar{y}z]$ [m → min terms]

SOP: 3 (011), 5 (101), 7 (111), 1 (001)

Missing terms: 0 (000), 2 (010), 4 (100), 6 (110)

POS: $\pi(m_0\cdot m_2\cdot m_4\cdot m_6)$

∴ $F \Rightarrow \pi[(x+y+z)\cdot(x+\bar{y}+z)\cdot(\bar{x}+y+z)\cdot(\bar{x}+\bar{y}+z)]$

b) $F = A + BC$

Sol: $A(B+B')(C+C') + BC(A+A')$

$\Rightarrow (AB+AB')(C+C') + ABC + A'BC$

$\Rightarrow \underline{ABC} + AB'C + ABC' + AB'C' + \underline{ABC} + A'BC$

$\Rightarrow ABC + A'BC + AB'C + ABC' + AB'C'$.

* Short hand notation:

$F = m_7 + m_3 + m_5 + m_6 + m_4$

$\Rightarrow \sigma(7,6,5,4,3)$.

c) $F = \bar{Y} + \bar{X}\bar{Z}$.

Sol: $\Rightarrow Y'(X+X')(Z+Z') + X'Z'(Y+Y')$

$\Rightarrow (XY' + X'Y')(Z+Z') + X'YZ' + X'Y'Z'$

A	B	C	D	Term	Max Term
0	0	0	0	$A+B+C+D$	m_0
0	0	0	1	$A+B+C+\bar{D}$	m_1
0	0	1	0	$A+B+\bar{C}+D$	m_2
0	0	1	1	$A+B+\bar{C}+\bar{D}$	m_3
0	1	0	0	$A+\bar{B}+C+D$	m_4
0	1	0	1	$A+\bar{B}+C+\bar{D}$	m_5
0	1	1	0	$A+\bar{B}+\bar{C}+D$	m_6
0	1	1	1	$A+\bar{B}+\bar{C}+\bar{D}$	m_7
1	0	0	0	$\bar{A}+B+C+D$	m_8
1	0	0	1	$\bar{A}+B+C+\bar{D}$	m_9
1	0	1	0	$\bar{A}+B+\bar{C}+D$	m_{10}
1	0	1	1	$\bar{A}+B+\bar{C}+\bar{D}$	m_{11}
1	1	0	0	$\bar{A}+\bar{B}+C+D$	m_{12}
1	1	0	1	$\bar{A}+\bar{B}+C+\bar{D}$	m_{13}
1	1	1	0	$\bar{A}+\bar{B}+\bar{C}+D$	m_{14}
1	1	1	1	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$	m_{15}

* Short Hand Notation:

$\pi(m_0, m_1, m_2, m_8, m_4, m_5, m_9, m_{10}, m_{11}, m_3)$

⊖ Max terms (POS):

a) $(x+y)\cdot(x'+z)$

Sol: $[(x+y)+(z\cdot z')]\cdot[(x'+z)+(y\cdot y')]$

$\Rightarrow [(x+y+z)\cdot(x+y+z')]\cdot[(x'+y+z)\cdot(x'+y'+z)]$

b) $(\bar{A}+B)\cdot(\bar{B}+C)$.

Sol: $[(\bar{A}+B)+(C\cdot C')]\cdot[(\bar{B}+C)+(A\cdot A')]$

$\Rightarrow (A'+B+C)\cdot(A'+B+C')\cdot(A+B'+C)\cdot(A'+B'+C)$.

⊖ Min terms:

a) $F = A\cdot[B+C(AB+AC)]$.

Sol: $A\cdot[B+C(AB+AC)]$

$\Rightarrow A\cdot[B+C\cdot AB+AC]$ $[\because C\cdot C=C]$

$\Rightarrow A\cdot B+ABC+AC$

$\Rightarrow AB(C+C')+ABC+AC(B+B')$

$\Rightarrow \underline{ABC}+ABC'+\underline{ABC}+\underline{ABC}+AB'C$

$\Rightarrow ABC+ABC'+AB'C$.

* In SOP,

A	B	C	Term	Minterms
0	0	0	$\bar{A}\cdot\bar{B}\cdot\bar{C}$	m_0
0	0	1	$\bar{A}\cdot\bar{B}\cdot C$	m_1
0	1	0	$\bar{A}\cdot B\cdot\bar{C}$	m_2
0	1	1	$\bar{A}\cdot BC$	m_3
1	0	0	$A\cdot\bar{B}\bar{C}$	m_4
1	0	1	$A\cdot\bar{B}\cdot C$	m_5
1	1	0	$AB\bar{C}$	m_6
1	1	1	$A\cdot B\cdot C$	m_7

∴ Short hand notation:

$F = m_7+m_6+m_5$

$\Rightarrow \sigma(m_7, m_6, m_5)$

$\Rightarrow \sigma(7,6,5)$

* POS to Canonical POS:

* $F = (p+q)\cdot(p+r)\cdot(q+r)$. $[A\cdot A' = 0]$

Sol: $[(p+q)+(r\cdot r')]\cdot[(p+r)+(q\cdot q')]\cdot[(q+r)+(p\cdot p')]$

$\Rightarrow [(p+q+r)\cdot(p+q+r')]\cdot[(p+q+r)\cdot(p+q'+r)]$
$\cdot[(p+q+r)\cdot(p'+q+r)]$ $[A\cdot A = A]$

$\Rightarrow (p+q+r)\cdot(p+q+r')\cdot(p+q'+r)\cdot(p'+q+r)$:

* Canonical POS to POS:

* $F = (p+q+r)\cdot(p+q+r')\cdot(p+q'+r)\cdot(p'+q+r)$.

Sol: $(p+q+r)(p+q+r)(p+q+r)\cdot(p+q+r')\cdot(p+q'+r)$
$\cdot(p'+q+r)$ $[A = A\cdot A]$.

$\Rightarrow [(p+q+r)\cdot(p+q+r')]\cdot[(p+q+r)\cdot(p+q'+r)]$
$\cdot[(p+q+r)\cdot(p'+q+r)]$ $[0 = A\cdot\bar{A}]$.

$\Rightarrow [(p+q)+(r\cdot r')]\cdot[(p+r)+(q\cdot q')]+[(p\cdot p')+(q+r)]$

$\Rightarrow (p+q)\cdot(p+r)\cdot(q+r)$.

⊙ Min terms:

a) $F = x'z' + y'z' + yz' + xy$ (SOP).

Sol: $x'z'(y+y') + y'z'(x+x') + yz'(x+x') + xy(z+z')$

$\Rightarrow x'z'y + x'z'y' + xy'z' + x'y'z' + xyz' + x'z'y + xyz$
$+ xyz'$.

$\Rightarrow x'z'y' + x'z'y + xyz + xy'z' + xyz'$.

b) $F = AC + B'D + A'CD + ABCD$

Sol: $\Rightarrow AC(B+B')(D+D') + B'D(A+A')(C+C')$
$+ A'CD(B+B') + ABCD$

$\Rightarrow (ACB + ACB')(D+D') + (AB'D + A'B'D)(C+C')$
$+(A'BCD + A'B'CD) + ABCD$.

$\Rightarrow ABCD + AB'CD + ABCD' + AB'CD' + AB'CD + AB'C'D$
$+ A'B'CD + A'B'C'D + A'BCD + A'B'CD + ABCD$.

$\Rightarrow ABCD + A'B'CD + A'B'C'D + AB'CD + ABCD' + AB'CD'$
$+ AB'C'D + A'BCD$.

* In SOP,

A	B	Term	Minterms
0	0	$\bar{A}\cdot\bar{B}$	m_0
0	1	$\bar{A}\cdot B$	m_1
1	0	$A\cdot\bar{B}$	m_2
1	1	$A\cdot B$	m_3

• Ex: $F = \bar{A}\bar{B} + AB$

$\Rightarrow \Sigma(m_0 + m_3) \Rightarrow \Sigma(0,3) \rightarrow$ Active

& $F' = \bar{A}\cdot B + A\cdot\bar{B}$ (Inverse)

$\Rightarrow \Sigma(m_1 + m_2) \Rightarrow \Sigma(1,2) \rightarrow$ Active

* In POS,

A	B	Term	Max-terms
0	0	$A+B$	m_0
0	1	$A+\bar{B}$	m_1
1	0	$\bar{A}+B$	m_2
1	1	$\bar{A}+\bar{B}$	m_3

• Ex: $F = (A+B)\cdot(\bar{A}+\bar{B})$

$\Rightarrow \pi(m_0 \cdot m_3) = \pi(0,3)$

& $F' = (A+\bar{B})\cdot(\bar{A}+B)$. [Inverse]

$\Rightarrow \pi(m_1 \cdot m_2) \Rightarrow \pi(1,2)$.

⊳ SOP to Canonical SOP:

* $F = pq + pr + qr$.

Sol: $F = pq(r+r') + pr(q+q') + qr(p+p')$ $(1 = A + A')$

$\Rightarrow pqr + pqr' + pqr + prq' + pqr + p'qr$.

$\Rightarrow pqr + pqr' + pq'r + p'qr$. $(A+A=A)$

⊳ Canonical SOP to SOP:

* $F = pqr + pqr' + pq'r + p'qr$.

Sol: $pqr + pqr' + pqr + pq'r + pqr + p'qr$ $(A = A+A)$

$\Rightarrow pq(r+r') + pr(q+q') + qr(p+p')$

$\Rightarrow pq + pr + qr$. $(A+A'=1)$

* $F = \bar{A}B(A+\bar{B})(\bar{B}+B)$.

$\Rightarrow \bar{A}B(A+\bar{B})$

$\Rightarrow A\bar{A}B + \bar{A}B\bar{B}$ $\quad [\because A \cdot A' = 0]$

$\Rightarrow 0+0 = 0$

$\therefore F = 0$

Standard form: Each term doesn't have all literals

Canonical forms: (To simplify the Boolean expressions) each term should have all literals

For a Boolean Expression, there are two kinds of Canocial forms:

1. Sum of products / Sum of min term

⇒ A min. term is a product of all variables taken either in direct or complemented form.

⇒ Any boolean func can be expressed as a sum of its 1 min. terms & the inverse of the func can be expressed as 0 min. terms.

2. Product of Sums / Product of Max terms.

⇒ A max term is a addition of all variables taken either in direct or complemented form.

⇒ Any boolean func can be expressed as a product of 0 max terms & the inverse of the func can be expressed as 1 max terms.

In

Sum of products, (SOP)

- F(list of variables) = Σ(list of min. term indices).

&

Inverse

- F'(list of variables) = Σ (list of 0 min term indices).

In product of Sums, (POS)

- F (list of variables) = π (list of 0 max term indices)

&

- F'(list of variables) = π (list of 1 max term indices).

* $F = A'B' + AC' + B'C'$.

Sol: $A'B' + AC' + B'C'(A + \bar{A})$

$\Rightarrow A'B' + AC' + AB'C' + A'B'C'$

$\Rightarrow A'B'(1 + C') + C'(A + AB')$

$\Rightarrow A'B'(1) + C'(A + AB')$

$\Rightarrow A'B' + C'(A)(1 + B')$

$\Rightarrow AB' + AC'$

$\therefore F = A'B' + AC'$

* $F = AB + B\bar{C} + AC$.

Sol: $AB + B\bar{C} + AC$ $(C + C')$

$\Rightarrow AC + B\bar{C} + B \cdot AC + AC'B$

$\Rightarrow B\bar{C}(1 + A) + AC(1 + B)$ $[1 + A = 1, \; 1 + B = 1]$

$\Rightarrow B\bar{C} + AC$

$\therefore F = AC + B\bar{C}$

* $F = A\bar{B} + BC + AC$.

Sol: $A\bar{B} + BC + AC(B + \bar{B})$

$\Rightarrow A\bar{B} + BC + ABC + AC\bar{B}$

$\Rightarrow BC(1 + A) + A\bar{B}(1 + C)$

$\Rightarrow BC + A\bar{B}$.

$\therefore F = A\bar{B} + BC$

* $F = x'yz + x'yz' + xz$.

Sol: $x'y(z + z') + xz$

$\Rightarrow x'y + xz$.

$\therefore F = x'y + xz$

* $F = A + A \cdot B$.

Sol: $A(1 + B) = A$.

$\therefore F = A$

* $(A+B)\cdot(A+C)$.

Sol: $(A+B)\cdot A + (A+B)\cdot C$

$\Rightarrow 1\cdot A + C\cdot A + C\cdot B$

$\Rightarrow A\cdot(1+C) + CB \Rightarrow A + A\cdot C + C\cdot B$

$\Rightarrow A + BC$

$\therefore (A+B)\cdot(A+C) = A+BC$

(or) $(A+C)\cdot A + B\cdot(A+C)$

$\Rightarrow A + A\cdot B + B\cdot C$

$\Rightarrow A + BC$

* $F = \bar{A}(A+B) + (B+A\cdot A)\cdot(A+\bar{B})$.

Sol: $A\cdot\bar{A} + A'\cdot B + (B+A)\cdot(A+\bar{B})$

$\Rightarrow A'\cdot B + A\cdot(A+B) + \bar{B}\cdot(A+B)$ $(A\cdot A' = 0)$

$\Rightarrow A + A'B + A\cdot\bar{B} + B\cdot B'$ $(B\cdot B' = 0)$

$\Rightarrow A + A'B + AB'$

$\Rightarrow A(1+B') + A'B$ $(1+B' = 1)$

$\Rightarrow A + A'B$

$\Rightarrow A+B$ (Simplification law).

(or).

$\bar{A}(A+B) + (A+B)(A+\bar{B})$

$\Rightarrow A+B(\bar{A}+A+\bar{B})$

$\Rightarrow A+B(1+\bar{B}) = A+B$.

$\therefore F = A+B$.

* $F = AB + A'C + BC$.

Sol: $AB + A'C + BC\cdot 1$

$\Rightarrow AB + A'C + BC\cdot(A+\bar{A})$

$\Rightarrow AB + A'C + BC\cdot A + BC\cdot A'$

$\Rightarrow AB + ABC + A'(C+BC)$

$\Rightarrow AB + ABC + C\cdot A'$

$\Rightarrow AB(1+C) + A'C$ $(1+C = 1)$

$\Rightarrow AB + A'C$.

$\therefore F = AB + A'C$.

* Solve the expression : $F = AB + AB'C + AB'C'$.

Sol: $AB + AB'(C+C')$

$\Rightarrow AB + AB'(1)$

$\Rightarrow A \cdot (B+B') = A \cdot 1 = A$.

A	B	C	B'	C'	A·B	A·B'	(AB')C	(AB')·C'	F
0	0	0	1	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0
1	0	0	1	1	0	1	0	1	1
1	0	1	1	0	0	1	1	0	1
1	1	0	0	1	1	0	0	0	1
1	1	1	0	0	1	0	0	0	1

$\therefore F = A$

* Solve the expression $F = A'B + BC' + BC + AB'C'$.

Sol: $F = A'B + BC' + BC + AB'C' + BC'$

$\Rightarrow A'B + B(C+C') + AB'C' + BC'$ $\quad (\because BC' = BC' + BC')$

$\Rightarrow A'B + B(1) + AB'C' + BC'$

$\Rightarrow B + A'B + C'(B + AB')$

$\Rightarrow A'B + B + C'(B+A)$

$\Rightarrow B(A'+1) + C'(B+A)$

$\Rightarrow B + C' \cdot B + AC'$ $\quad (1 + A' = 1)$

$\Rightarrow B(1+C') + AC'$

$\Rightarrow B + AC'$

$\therefore F = B + AC'$

ii)

A	B	C	B+C	A·(B+C)	A·B	A·C	(A·B)+(A·C)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

$\therefore A\cdot(B+C) = (A\cdot B)+(A\cdot C)$.

10.

i)

A	B	$\sim A$	$\sim B$	A·B	$\sim(A\cdot B)$	$(\sim A)+(\sim B)$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

$\therefore \sim(A\cdot B) = (\sim A)+(\sim B)$.

* Solve the following Boolean expression : $F = AB + AB'$.

Sol: $F = AB + AB'$

$\Rightarrow A\cdot(B+B')$

$\Rightarrow A\cdot(1) = A$

cross check using truth table

$\therefore F = AB + AB' = A$.

ii)

A	B	$\sim B$	A·B	A·($\sim$B)	AB + AB'
0	0	1	0	0	0
0	1	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1

$\therefore AB + AB' = A$

▷ Boolean / Algebraic Identities:

- It is a set of rules used to simplify the Boolean expressions.
- These are used when list of variables are less.

1. Double Complement law: $\sim(\sim A) = A$.
2. Complement law: (i) $A + (\sim A) = 1$ (OR form)
   (ii) $A \cdot (\sim A) = 0$ (AND form).
3. Idempotent law: (i) $A + A = A$ (OR form)
   (ii) $A \cdot A = A$ (AND form)
4. Dominance law: (i) $A + 1 = 1$ (OR form)
   (ii) $A \cdot 0 = 0$ (AND form).
11. Identity law:
   - $A + 0 = A$
   - $A \cdot 1 = A$
5. Commutative law: (i) $A + B = B + A$ (OR form)
   (ii) $A \cdot B = B \cdot A$ (AND form)
6. Associative law: (i) $A + (B + C) = (A + B) + C$ (OR).
   (ii) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ (AND).
7. Absorption law: (i) $A \cdot (A + B) = A$
   (ii) $A + (A \cdot B) = A$.

   $B + AB' = B + A$.
8. Simplification law: (i) $A \cdot (\sim A + B) = A \cdot B$
   (ii) $A + (\sim A \cdot B) = A + B$.

   $1 + A' = 1$
9. Distributive law: (i) $A + (B \cdot C) = (A + B) \cdot (A + C)$
   (ii) $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$
10. De-Morgan's law: (i) $\sim(A \cdot B) = (\sim A) + (\sim B)$
    (ii) $\sim(A + B) = (\sim A) \cdot (\sim B)$

3 variable
$2^3 = 8$ values
2 variables
$2^2 = 4$ values

9.

* i)

A	B	C	B·C	A+(B·C)	A+B	A+C	(A+B)·(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

$\therefore A + (B \cdot C) = (A + B) \cdot (A + C)$

∴ Corrected Code : 1001100

* For Odd Parity,

Ex: 1001.

0		1	1			
1	0	0	P_4	1	P_2	P_1
7	6	5	4	3	2	1

- $P_1 = (3,5,7) = 1$ (1's (E))
  1 0 1
- $P_2 = (3,6,7) = 1$ (1's (E))
  1 0 1
- $P_4 = (5,6,7) = 0$ (1's (0))
  0 0 1

∴ Hamming Code = 1000111

- Error Code: 0000111
  - $E_1 = (1,3,5,7) = 1$
    1 1 0 0
  - $E_2 = (2,3,5,7) = 1$ } 1's (E)
    1 1 0 0
  - $E_3 = (4,5,6,7) = 1$
    0 0 0 0

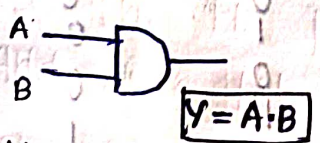
∴ 111 = 7th position of Error.

∴ Corrected Code : 1000111

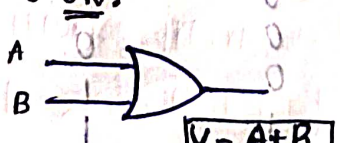
⊳ Logic Gates: AND OR NOT NAND NOR XOR XNOR

A	B	$\bar{A}$	$\bar{B}$	$A \cdot B$	$A+B$	$\overline{A \cdot B}$	$\overline{A+B}$	$A \oplus B$	$A \odot B$
0	0	1	1	0	0	1	1	0	1
0	1	1	0	0	1	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	1	0	0	1	1	0	0	0	1

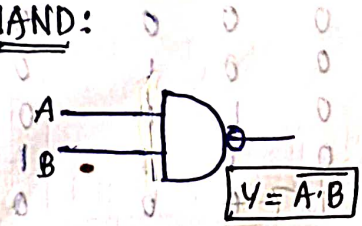
- AND:



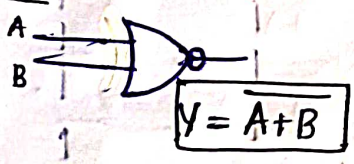
- OR:



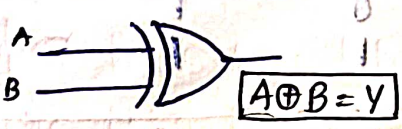
- NAND:



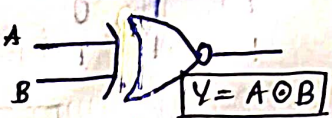
- NOR:



- XOR



- XNOR:



- NOT:

(In general, we req. to solve even PB).

- Set a PB to 0, if the total no. of 1's count is even.
- The total no. of PB's added to our original message is described as $2^P \geq P + m + 1$

  where $P \rightarrow$ PB's

  $m \rightarrow$ message / data bit

⇒ Ex: 1001 (4-bit) ; $m = 4$ (For Even Parity).

$2^P \geq P + 4 + 1 \Rightarrow P = 3$

- Note: The min. no. of PB's req. for 4-bit message ⇒ 3.
- Hamming Code Algorithm for 4-bit message is represented by $(7,4)$.

Sol:

1 0 0 1 1 0 0 ← original message bits

D_7	P_6	D_5	P_4	D_3	P_2	P_1
7	6	5	4	3	2	1

[Even 1's - Even - 0, Odd - 1]

- $P_1 = (1, 3, 5, 7)$ → 1 0 1 (no. of 1's = 2(E)) PB = 0
- $P_2 = (2, 3, 6, 7)$ → 1 0 1 (1's = 2(E)) PB = 0
- $P_4 = (4, 5, 6, 7)$ → 0 0 1 (1's = 1(O)) PB = 1

∴ Hamming Code = 1001100 → Encoded message

- To encode the original message, PB's are added.
- No. of error bits are equal to the no. of PB's added.

Eg: $E_1 = (1,3,5,7) = 0$ → 0 1 0 1 → 1's (E)

$E_2 = (2,3,6,7) = 1$ → 0 1 1 1 → is (O)

$E_3 = (4,5,6,7) = 1$ → 1 0 1 1 → 1's (O)

- Error Code: 1101100 → Error position

1	0	0	E_3	1	E_2	E_1
7	6	5	4	3	2	1

Error detection = 6 (110) → position of Error.

detect & correct the codes that occurs when the data is moved/stored from sender to the reciever.

⊙ Alphanumeric Codes:

- It is the combination of alphabets, numbers, special characters and signals.

* Hamming Code Technique: (Single-bit error code).

- It is a set of error correction codes.

⇒ Parity Bit: It is a bit appended to the original data to ensure that total no. of 1's in the data are even or odd.

- There are 2 types of parity bits
  1. Even parity.
  2. Odd parity.

* In even parity, if no. of 1's are even, then parity bit will be 0 & if no. of 1's are odd in no, then parity bit will be 1.

* Ex: 11100 ; no. of 1's = 3 (odd)
  So, parity bit = 1

* In odd parity, if no. of 1's are even, then PB = 1 & if odd, PB = 0.

- Ex: 11100 ; no. of 1's = 3 (o)
  PB = 0.

⇒ General Algorithm:

- Write the bit positions from 1
- Set the PB positions from 2^0 to 2^n
- Add data bits at remaining positions
- Set PB covers; $2^0 = 1$ (check 1 bit & skip 1 bit) → (1, 3, 5, ... skip 2, 4)
  $2^1 = 2$ (check 2 bits & skip 2) → (2, 3, 6, 7) skip 4, 5, ...
  $2^2 = 4$ (check 4 & skip 4) → (4, 5, 6, 7) skip 8, 9, 10, 11, ...
- Set a PB to 1, if the total no. of 1's count is odd.

DC	Gray	BCD
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001

▷ BCD (Binary Coded Decimal) : It is a way to represent or express decimal number system in which each digit is represented with 4-bit Binary group.

▷

• Binary	• BCD
* Length is limitless	* It length should be 4 bits.
* We can do it for all values.	* 0-9 combinations are valid & remaining are invalid (10-15)
	* It is similar to decimal number system.

* ASCII - American Standard Code for information Interchange.

▷ Error Correction / Detection: (It is a condition when output information doesn't match with the input info. To avoid this, we use error detecting codes. ("Hamming code technique").

* It is developed by R.W. Hamming. It is a set of error correction codes that can be used to

* Gray: (Reflected binary code)

X-OR:

A	B	$A\bar{B}+\bar{A}B$
0	0	0
1	0	1
0	1	1
1	1	0

* Two successive values differ only in 1 bit.

DC	BCD.	Gray.
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1

⇒ Note:

i) original code: $b_3\ b_2\ b_1\ b_0$ (performed in reverse).

most significant bit ↓ ... ↖ least significant bit

Gray Code: $g_3\ g_2\ g_1\ g_0$

$$\begin{cases} g_3 = b_3 \\ g_2 = b_2 \oplus b_3 \\ g_1 = b_1 \oplus b_2 \\ g_0 = b_0 \oplus b_1 \end{cases}$$

ii) Gray Code: $g_3\ g_2\ g_1\ g_0$ (stays the same)

↓

Binary Code: $b_3\ b_2\ b_1\ b_0$ (stays the same)

$$\begin{cases} b_0 = b_1 \oplus g_0 \\ b_1 = b_2 \oplus g_1 \\ b_2 = b_3 \oplus g_2 \\ b_3 = g_3 \end{cases}$$

* Ex: 0 0 1 1

0 0 1 0

Non-Weighted Codes: (No principle).

Eg: i) 0 0 1 0 (2)
+ 0 1 1 0 (6)
1 0 0 0 (8)

(Derived from BCD 8421)
• Eg: XS-3, Gray
(It is used to express decimal values).

⇒ Binary Addition: * A B A+B
0 0 0
1 0 1
0 1 1
1 1 0 with carry 1.

ii) 1 1 1
0 1 1 1 (7)
+ 0 0 1 1 (3)
1 0 1 0 (10)

* XS-3:

DC	BCD 8 4 2 1	+3 (+0011)	Non-weighted (XS-3).
0	0 0 0 0		0 0 1 1 (3)
1	0 0 0 1		0 1 0 0 (4)
2	0 0 1 0		0 1 0 1 (5)
3	0 0 1 1		0 1 1 0 (6)
4	0 1 0 0		0 1 1 1 (7)
5	0 1 0 1		1 0 0 0 (8)
6	0 1 1 0		1 0 0 1 (9)
7	0 1 1 1		1 0 1 0 (10) - A
8	1 0 0 0		1 0 1 1 (11) - B
9	1 0 0 1		1 1 0 0 (12) - C

* 0: 0000
+ 0011 (3)
0011 (3)

* 1: 0001
0011 (3)
0100 (4)

* 2: 0010
0011 (3)
0101 (5)

* 3: 0011
0011 (3)
0110 (6)

* 4: 0100
0011 (3)
0111 (7)

* 5: 0101
0011 (3)
1000 (8)

* 6: 0110
0011 (3)
1001 (9)

* 7: 0111
0011 (3)
1010 (10)
A

* 8: 1000
0011 (3)
1011 (11)
B

* 9: 1001
0011 (3)
1100 (12)
C

▷ Weighted Codes: (obeys positional weight principle)

Ex: 2421, 5421, 8421, 5211

* DC

BCD

	8	4	2	1	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	2+1=3
4	0	1	0	0	4
5	0	1	0	1	4+1=5
6	0	1	1	0	4+2=6
7	0	1	1	1	4+2+1=7
8	1	0	0	0	⇒ 8
9	1	0	0	1	→ 8+1=9

(1 - active, 0 - inactive)

* DC

BCD

	5	4	2	1	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	
4	0	1	0	0	
5	0	1	0	1	1000
6	1	0	0	1	0110
7	1	0	1	0	
8	1	0	1	1	
9	1	1	0	0	

* Each position represents a wt.

* DC

BCD

	2	4	2	1	
0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	
3	0	0	1	1	0010
4	1	0	0	1	0011
5	1	0	1	0	0100
6	1	0	1	1	0101
7	1	1	0	0	0110
8	0	1	1	1	1101
9	1	1	1	0	
	1	1	1	1	

$\therefore (46)_8 = (38)_{10}$

$= (100110)_2$

$= (26)_{16}$.

4. $(1010)_2 =$

Sol: i) $(1010)_2 = (\quad)_{10}$

$1\ 0\ 1\ 0$

$2^3\ 2^2\ 2^1\ 2^0$

$\Rightarrow 0 + 2 + 0 + 8 = (10)_{10}$

ii) $(1010)_2 = (\quad)_8$

$001\ 010$

$(1\quad 2)_8$

iii) $(1010)_2 = (\quad)_{16}$

$\Rightarrow (A)_{16}$

$\therefore (1010)_2 = (10)_{10}$

$= (12)_8$

$= (A)_{16}$.

▷ Binary Codes:

- Group of symbols → Binary Code.
  - ↓ which on entering numbers, letters, Alphanumeric gets converted into.

⇒ Advantages:

- For digital communication
- Suitable for computer applications.
- Implementation becomes easy.

▷ Classification: Binary Codes

- Weighted codes
- Non weighted codes
- BCD (Binary coded Decimal)
- Alpha-Numeric
- Error correction / Detection

2. $(26)_{10}$ =

Sol: i) $(26)_{10}$ = ( )$_2$

$\Rightarrow$

2	26
2	13 - 0
2	6 - 1
2	3 - 0
2	1 - 1
	0 - 1

$\Rightarrow (011010)_2$

ii) $(26)_2$ = ( )$_8$

2|6

$(010110)_8$.

iii) $(26)_2$ = ( )$_{16}$

2|6

$(0010\ 0110)_{16}$

$\therefore (26)_{10} = (011010)_2$

$(26)_2 = (010110)_8$

$(26)_2 = (00100110)_{16}$

vi) $(26)_{10}$ = ( )$_8$

8|26

3 - 2

$\Rightarrow (32)_8$

viii) $(26)_{10}$ = ( )$_{16}$

16| 26

1 - 10(A) $\Rightarrow (1A)_{16}$.

$\therefore (26)_{10} = (32)_8$

$= (1A)_{16}$

iv) $(26)_8$ = ( )$_{10}$

2 6

$8^1\ 8^0$ $\Rightarrow 16+6 = (22)_8$

v) $(26)_{16}$ = ( )$_{10}$

$\Rightarrow$ 2 6

$16^1\ 16^0$ $\Rightarrow 6+32 = (38)_{16}$

$\therefore (26)_8 = (22)_{10}$

$(26)_{16} = (38)_{10}$.

3. $(46)_8$ =

Sol: i) $(46)_8$ = ( )$_{10}$

4 6

$8^1\ 8^0$

$\Rightarrow 6\times8^0 + 4\times8 = 6+32 = (38)_{10}$.

ii) $(46)_8$ = ( )$_2$

4|6

$(100110)_2$

iii) $\Rightarrow (46)_8$ = ( )$_{16}$

0010|0110

(2 6)$_{16}$.

Number System:

They used to represent certain quantity or used to count discrete units. It has 4 categories:

1. Decimal (10).
2. Octal (8).
3. Binary (2).
4. Hexadecimal (16).

} Positional Value Systems. (i.e, the value of digits depends on its position).

⇒ **Conversions:** (To encrypt the data).

- Binary N.S → Easier to implement.
- Decimal N.S → Common.

1. $(ABCD)_{16}$.

Sol: i) $(ABCD)_{16} = (\quad)_{10}$

10 11 12 13

16^3 16^2 16^1 16^0

⇒ $13 \times 16^0 + 12 \times 16 + 11 \times 256 + 10 \times 4096$

⇒ $13 + 192 + 2816 + 40960$.

⇒ $(43981)_{10}$

11
40960
2816
192
13
43981

ii) $(ABCD)_{16} = (\quad)_2$

$(1010101111001101)_2$

iii) $(ABCD)_{16} = (\quad)_8$

⇒ 001 010 101 111 001 101

1 2 5 7 1 5.

⇒ $(125715)_8$.

∴ $(ABCD)_{16} = (43981)_{10}$
$= (1010101111001101)_2$
$= (125715)_8$.

1. DIGITAL COMPONENTS & DATA REPRESENTATION

⇒ Computer organisation is concerned with the structure & behaviour of the computer system. It deals with the components of a connection in a system.

⇒ It tells us how exactly all the units in the system are arranged & interconnected i.e, signals, adders & peripherals.

▷ Data: It refers to the symbols that represent people, events, things & ideas.

▷ Data Representation: It refers to the form in which data is stored, processed and transmitted. We can represent it in symbols, characters, text, no.'s, signals & special characters.

▷ Bits and Bytes:

⇒ Bits → Binary digit. It stores data in either 0 or 1.

⇒ 4 bits → Nibble.

8 bits → Byte.

⇒ 1024 Bytes = 1 KB

1024 KBytes = 1 MB.

1024 MB = 1 GB.

1024 GB's = 1 TB.

1024 TB's = 1 PB

1024 PB's = 1 EB. (Peta, Exa)

1024 EB's = 1 ZB. (Zeta)

▷ Memory Capacity Chart:

⇒ The [illegible] to describe the storage capacity &